Maharaja Education Trust (R), Mysuru

Maharaja Institute of Technology Mysore

Belawadi, Sriranga Pattana Taluk, Mandya – 571 477

Approved by AICTE, New Delhi,
Affiliated to VTU, Belagavi & Recognized by Government of Karnataka

Lecture Notes on

BIG DATA ANALYTICS (15CS82)

Prepared by

Department of Information Science and Engineering

Vision

"To be recognized as a premier technical and management institution promoting extensive education fostering research, innovation and entrepreneurial attitude"

Mission

- To empower students with indispensable knowledge through dedicated teaching and collaborative learning.
- To advance extensive research in science, engineering and management disciplines.
- To facilitate entrepreneurial skills through effective institute - industry collaboration and interaction with alumni.
- To instill the need to uphold ethics in every aspect.
- To mould holistic individuals capable of contributing to the advancement of the society.

VISION OF THE DEPARTMENT

To be recognized as the best centre for technical education and research in the field of information science and engineering.

MISSION OF THE DEPARTMENT

- To facilitate adequate transformation in students through a proficient teaching learning process with the guidance of mentors and all-inclusive professional activities.
- To infuse students with professional, ethical and leadership attributes through industry collaboration and alumni affiliation.
- To enhance research and entrepreneurship in associated domains and to facilitate real time problem solving.
-

PROGRAM EDUCATIONAL OBJECTIVES:

- Proficiency in being an IT professional, capable of providing genuine solutions to information science problems.
- Capable of using basic concepts and skills of science and IT disciplines to pursue greater competencies through higher education.
- Exhibit relevant professional skills and learned involvement to match the requirements of technological trends.

PROGRAM SPECIFIC OUTCOME:

Student will be able to

- **PSO1:** Apply the principles of theoretical foundations, data Organizations, networking concepts and data analytical methods in the evolving technologies.
- **PSO2:**Analyse proficient algorithms to develop software and hardware competence in both professional and industrial areas

Program Outcomes

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Course Overview

SUBJECT: BIG DATA ANALYTICS **SUBJECT CODE: 15CS82**

Big Data has been described by some Data Management pundits (with a bit of a snicker) as "huge, overwhelming, and uncontrollable amounts of information." In 1663, John Graunt dealt with "overwhelming amounts of information"

The Big Data course provides a unique approach to help students act on data for real business gain. The focus is not what a tool can do, but what you can do with the output from the tool. Gain the skills you need to store, manage, process, and analyze massive amounts of unstructured data to create an appropriate data lake.

As big data analytics is gaining popularity with every passing day, it is essential for businesses to be aware of the big data analytics predictions and stay abreast with all the latest trends. The companies will require **big data analytics** to work on these **data** to provide insight for the companies so student able to get a better job in companies.

Course Objectives

1. Understand Hadoop Distributed File system and examine MapReduce Programming
2. Explore Hadoop tools and manage Hadoop with Ambari
3. Appraise the role of Business intelligence and its applications across industries
4. Assess core data mining techniques for data analytics
5. Identify various Text Mining techniques

Course Outcomes

CO's	DESCRIPTION OF THE OUTCOMES
15CS82.1	Apply the basic concepts of warehousing, mining and Hadoop Distributed File System.
15CS82.2	Apply algorithms of warehousing, mining and Hadoop Distributed File System for data analysis.
15CS82.3	Analyze Regression, Clustering, Artificial Neural Networks and Decision tree techniques for decision making.
15CS82.4	Assess different techniques of Mining, Association and Mapreduce framework.

Maharaja Institute of Technology Mysore

Department of Information Science and Engineering

Syllabus

SUBJECT: BIG DATA ANALYTICS
SUBJECT CODE: 15CS82

Topics Covered as per Syllabus	Teaching Hours
MODULE-1:	
Hadoop Distributed File System Basics, Running Example Programs and Benchmarks, Hadoop MapReduce Framework, MapReduce Programming	**10 Hours**
MODULE-2:	
Essential Hadoop Tools, Hadoop YARN Applications, Managing Hadoop with Apache Ambari, Basic Hadoop Administration Procedures	**10 Hours**
MODULE -3:	
Business Intelligence Concepts and Application, Data Warehousing, Data Mining, Data Visualization	**10 Hours**
MODULE-4:	
Decision Trees, Regression, Artificial Neural Networks, Cluster Analysis, Association Rule Mining	**10 Hours**
MODULE-5:	
Text Mining, Naïve-Bayes Analysis, Support Vector Machines, Web Mining, Social Network Analysis	**10 Hours**

List of Text Books

T1. Douglas Eadline,"Hadoop 2 Quick-Start Guide: Learn the Essentials of Big Data Computing in the Apache Hadoop 2 Ecosystem", 1stEdition, Pearson Education, 2016. ISBN-13: 978-9332570351
T2. Anil Maheshwari, “Data Analytics”, 1st Edition, McGraw Hill Education, 2017. ISBN-13: 978-9352604180

List of Reference Books

T1. Tom White, “Hadoop: The Definitive Guide”, 4th Edition, O’Reilly Media, 2015.ISBN-13: 978-9352130672
T2. Boris Lublinsky, Kevin T.Smith, Alexey Yakubovich,"Professional Hadoop Solutions", 1stEdition, Wrox Press, 2014ISBN-13: 978-8126551071
T3. Eric Sammer,"Hadoop Operations: A Guide for Developers and Administrators",1stEdition, O'Reilly Media, 2012.ISBN-13: 978-9350239261.

List of URLs, Text Books, Notes, Multimedia Content, etc

1. https://www.tutorialspoint.com/big_data_tutorials.htm
2. https://nptel.ac.in/courses/106/104/106104189/

Index

SUBJECT BIG DATA ANALYTICS SUBJECT CODE: 15CS82

module -1
chapter -1

Hadoop Distributed file System Basics

Hadoop distributed file system was designed for Big data processing. It is capable of supporting many users simultaneously. The design of HDFS is based on the design of the Google File System (GFS).

HDFS is designed for data streaming where large amount of data are read from the disk in bulk. The HDFS size is typically 64 MB or 128 MB.

HDFS components

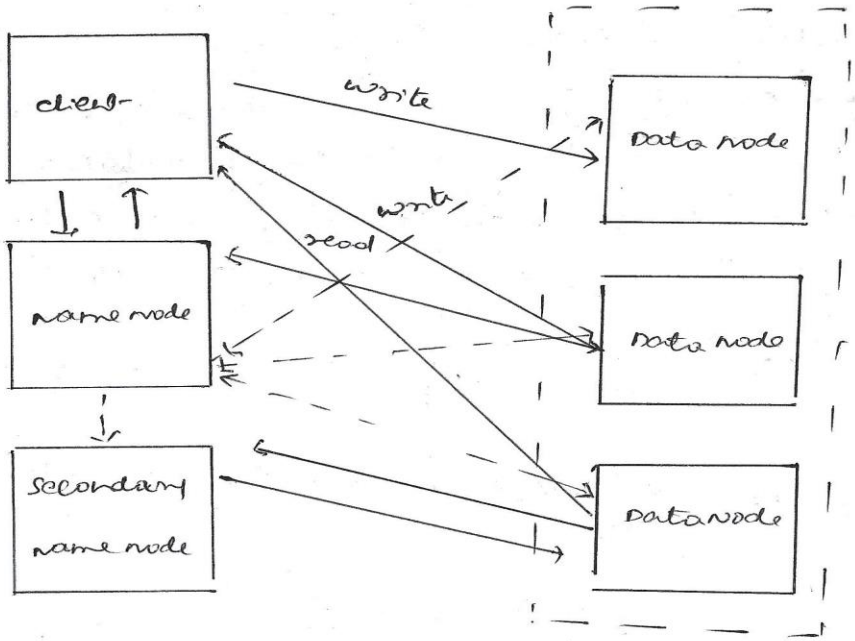
The design of HDFS is based on two types of nodes:

1) Name Node
2) Data Node

Name Node manages all the metadata needed to store and retrieve actual data from the Data Node. No data is actually stored on the Name Node.

master Name Node manages the file system namespace and regulates access to files by client. File system namespace operations such as opening, closing, and renaming files and directories are all managed by the namenode.

The slaves (Data Node) are responsible for serving read and write requests from the file system to the client. The name node manages block creation, deletion, and replication.



various system roles in an HDFS deployment.

Figure shows client / name node / Data node interactions.

When a client write data, it first communicates with the name node request to create a file. The name node determines how many blocks are needed and provides client with the Data node that will store the data.

The name node will attempt to write replicas of the data blocks on nodes that are in other separate block. If there is only one rack, then the replicated blocks are written to other server in the same rack. After the Data node acknowledges that the file block replication is complete, the client closes the file and inform the name node that the operation is complete.

Reading data happens in a similar fashion. (2) The client requests a file from the NameNode, which returns the best DataNode from which to read the data. The client access the data directly from the DataNode.

Once the metadata has been delivered to the client, the NameNode steps back and lets the conversation btw the client and the DataNode proceed. While the data transfer is progressing, the NameNode also monitors the DataNode by listening for heartbeats sent from DataNodes.

The purpose of the secondary name node is to perform periodic checkpoints that evaluate the status of the NameNode

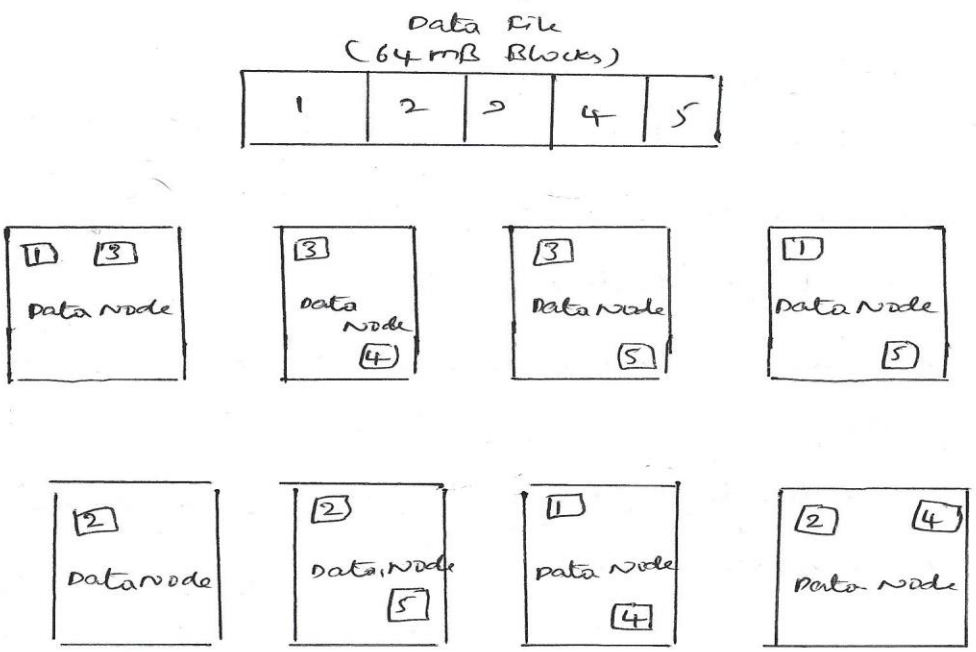
Various roles of HDFS

1. HDFS uses a master/slave model designed for large file reading/streaming
2. The NameNode is a metadata server.
3. HDFS provides a single namespace that is managed by the NameNode
4. HDFS provides a single namespace that is managed by the NameNode.
5. Data is redundantly stored on DataNode. There is no data on the NameNode
6. Secondary NameNode performs checkpoints of the NameNode file systems

HDFS Block Replication

When HDFS writes a file, it is replicated across the cluster. Hadoop cluster containing more than eight data nodes, the replication value is usually set to 3. Hadoop cluster of eight or fewer DataNode but more than one Data Node, a replication factor is 2. For a single machine replication factor is 1.

HDFS default block size is 64 mB. In typical OS the block size is 4 KB or 8 KB. The HDFS default block size is not the minimum block size. If a 20 KB file is written to HDFS it will create a block that is approximately 20 KB only. If a file size is 80 mB, a 64 mB block and a 16 mB block will be created.



HDFS block replication example

Figure provides an example of how a file is broken into blocks and replicated across the cluster. In this case, a replication factor of 3 ensures that any one data node can fail and the replicated blocks will be available on the other nodes — and then subsequent re-replicated on other data node.

HDFS safe mode

When the Name node starts, it enter a read-only safe mode where blocks can't be replicated or deleted. Safe mode enables the Name node to perform two important process.

1. The previous file system state is reconstructed by loading the fs image file into memory and replying the edit log.
2. The mapping between blocks and data nodes is created by waiting for enough of the Data node's to register so that at least one copy of the data is available. Not all the Data nodes are required to register before HDFS exists from safe mode. The registration process may continue for some time.

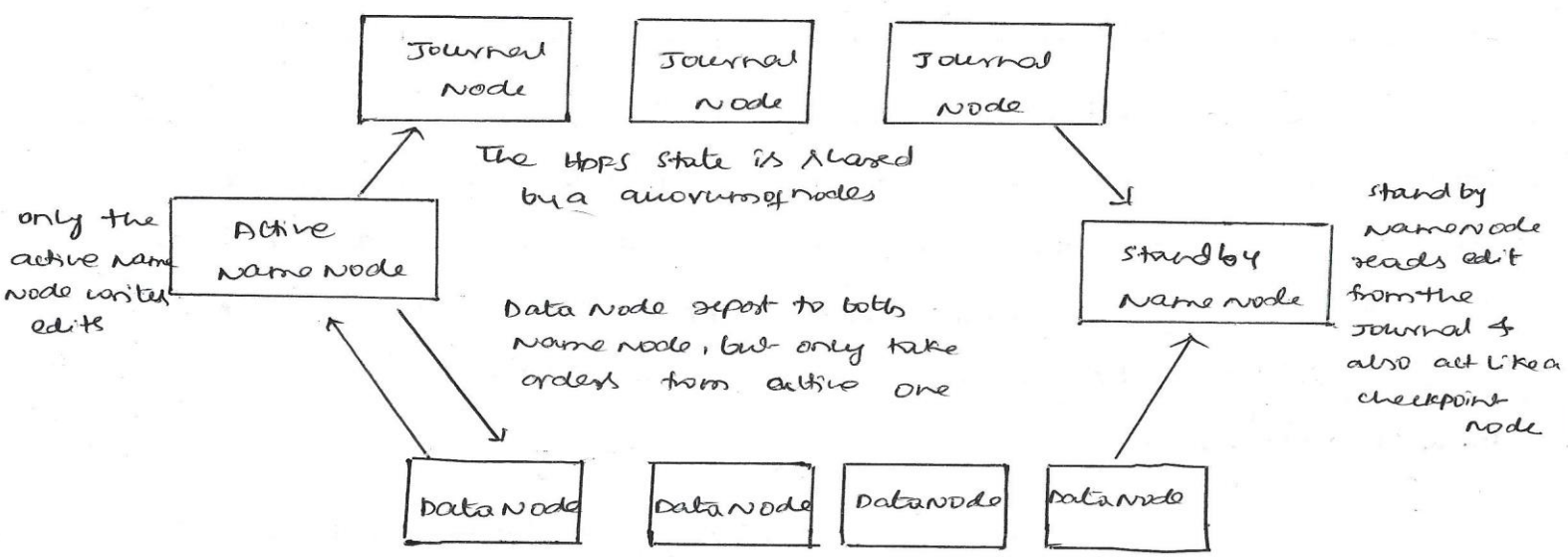
Rack Awareness

Rack awarenes deals with data locality. Recall that one of the main design goals of Hadoop MapReduce is to move the computation to the data. Assuming that most data center network does not offer full bisection bandwidth, a typical Hadoop cluster will exhibit three levels of data locality.

1. Data resides on the local machine (best)
2. Data resides on the same rack (better)
3. Data resides in a different rack (good).

Name node high Availability

The Name node was a single point of failure that could bring down the entire Hadoop cluster. Name node hardware often employed redundant power supplies and storage to guard against such problems, but it was still susceptible to other failure. The solution was to implement Name node high Availability (HA) as a means to provide true failover service.



HDFS high Availability design

An HA Hadoop cluster has two (or more) separate Name node machines. Each machine is configured with exactly the same software

One of the NameNode machines is in the Active state, and other is in the Standby state. Like a single NameNode cluster, the Active NameNode is responsible for all client HDFS operations in the cluster. The Standby NameNode maintains enough state to provide a fast failover.

To guarante the file system state is preserved, bothe the Active and Standby NameNode receive block reports from the DataNode. The Active node also sends all file system edit to a quorum of Journal nodes.

To prevent confusion between NameNode, the Journal nodes allow only one NameNode to be a writer at a time. During ~~failover~~ failover, the NameNode that is chosen to become active takes over the role of writing to the Journal nodes. A secondary NameNode is not required in the HA configuration because the standby node also perform the task of the secondary NameNode.

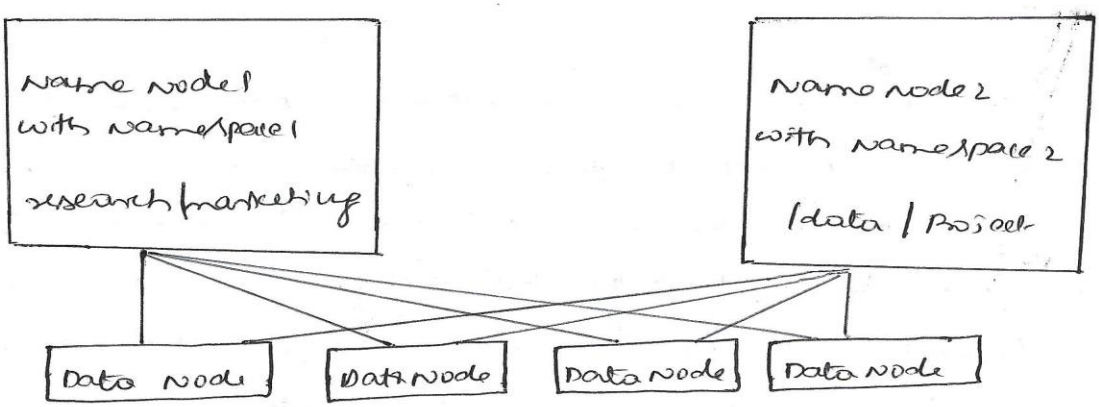
HDFS NameNode Federation

Another important feature of HDFS is NameNode Federation. HDFS provided a single name space for the entire cluster managed by a single NameNode. Thus resource of a single NameNode determine the size of the name space. The key benefits are as follows

- Namespace scalability :- HDFS cluster storage scales horizontally without placing a burden on the NameNode.

- Better performance :- Adding more name node to the cluster scales the file system read/write operations throughput by separating the total name space.

- System isolation :- multiple name nodes enable different categories of applications to be distinguished, and user can be isolated to different name space.



HDFS Name Node federation

Figure illustrates how HDFS Name Node federation is accomplished. Name Node1 manages the /research and /marketing name spaces and Name node2 manages the /data & /project name space. The Name Node don't communicate with each other and the Data Node1 " just store data blocks" as directed by either Name Node.

(5)

HDFS checkpoints and Backup

Name Node stores the metadata of the HDFS file system in a file called fsimage. File system modifications are written to an edit log file and at startup the Name Node merges the edits into a new fsimage. The secondary Name Node or checkpoint Node periodically fetches edits from the Name Node, merge them and returns an updated fsimage to the Name Node.

An HDFS Backup Node is similar, but also maintains an up-to-date copy of the file system name space both in memory and on disk. A Name Node support one Backup node at a time. No checkpoint nodes may be registered if a Backup node is in use.

HDFS Snapshots

HDFS snapshots are similar to backups, but are created by administrators using the hdfs dfs -snapshot command. HDFS snapshots are read-only point-in-time copies of the file system. They offer following features.

1. Snapshots can be taken of a sub-tree of the file system or the entire file system.
2. Snapshots can be used for data backup, protection against user errors, and disaster recovery.
3. Snapshots creation is instantaneous.

4. Blocks on the data node are not copied, because the snapshot files recorded the block list and the file size.

5. Snap shots do not adversely affect regular HDFS operations.

HDFS NFS Gateway

The HDFS NFS Gateway supports NFSV3 and enables HDFS to be mounted as part of the client's local file systems. Users can browse the HDFS file system through their local file system that provide an NFSV3 client compatible OS. This feature offers users the following capabilities:

1. User can easily download / upload files from / to the HDFS file system to/from their local file system.

2. User can stream data directly to HDFS through the mount point. Appending to a file is supported, but random write capability is not supported.

MODULE 1
Chapter 2: Running Example program and Bench Mark

When using new or updated hardware or software, simple examples and benchmarks help confirm proper operation. Apache Hadoop includes many examples and bench- marks to aid in this task. This chapter provides instructions on how to run, monitor, and manage some basic Map Reduce examples and benchmarks

Running Map Reduce Examples

All Hadoop releases come with MapReduce example applications. Running the existing MapReduce examples is a simple process—once the example files are located, that is. For example, if you installed Hadoop version 2.6.0 from the Apache sources under
/opt, the examples will be in the following directory:
/opt/hadoop-2.6.0/share/hadoop/mapreduce/

In other versions, the examples may be in /usr/lib/hadoop-mapreduce/ or some other location. The exact location of the example jar file can be found using the find command:
$ find / -name "hadoop-mapreduce-examples*.jar" -print

For this chapter the following software environment will be used:
OS: Linux
Platform: RHEL 6.6
Hortonworks HDP 2.2 with Hadoop Version: 2.6
In this environment, the location of the examples is /usr/hdp/2.2.4.2-2/hadoop- mapreduce. For the purposes of this example, an environment variable called HADOOP_EXAMPLES can be defined as follows:
$ export HADOOP_EXAMPLES=/usr/hdp/2.2.4.2-2/hadoop-mapreduce

Once you define the examples path, you can run the Hadoop examples using the commands discussed in the following sections.
Business intelligence (BI) is an umbrella term that includes a variety of IT applications that are used to analyze an organization’s data and communicate the information to relevant users.

Listing Available Examples

A list of the available examples can be found by running the following command. In some cases, the version number may be part of the jar file (e.g., in the version 2.6 Apache sources, the file is named hadoop-mapreduce-examples-2.6.0.jar).

$ yarn jar $HADOOP_EXAMPLES/hadoop-mapreduce-examples.jar

Note

In previous versions of Hadoop, the command hadoop jar . . . was used to run MapReduce programs. Newer versions provide the yarn command, which offers more capabilities. Both commands will work for these examples.

The possible examples are as follows:

An example program must be given as the first argument. Valid program names are:

aggregatewordcount: An Aggregate based map/reduce program that counts the words in the input files.

aggregatewordhist: An Aggregate based map/reduce program that computes the histogram of the words in the input files.

bbp: A map/reduce program that uses Bailey-Borwein-Plouffe to compute exact digits of Pi.

dbcount: An example job that count the pageview counts from a database. distbbp: A map/reduce program that uses a BBP-type formula to compute exact bits of Pi.

grep: A map/reduce program that counts the matches of a regex in the input.

join: A job that effects a join over sorted, equally partitioned datasets

multifilewc: A job that counts words from several files. pentomino: A map/reduce tile laying program to find solutions to pentomino problems.

pi: A map/reduce program that estimates Pi using a quasi-Monte Carlo method.

randomtextwriter: A map/reduce program that writes 10GB of random textual data per node.

randomwriter: A map/reduce program that writes 10GB of random data per node.

secondarysort: An example defining a secondary sort to the reduce. sort: A map/reduce program that sorts the data written by the random writer.

sudoku: A sudoku solver.

teragen: Generate data for the terasort terasort: Run the terasort

teravalidate: Checking results of terasort

wordcount: A map/reduce program that counts the words in the input files.
wordmean: A map/reduce program that counts the average length of the words in the input files.
wordmedian: A map/reduce program that counts the median length of the words in the input files.
Word standard deviation: A map/reduce program that counts the standard deviation of the length of the words in the input files.

To illustrate several features of Hadoop and the YARN Resource Manager service GUI, the pi and terasort examples are presented next.

Running the Pi Example

The pi example calculates the digits of using a quasi-Monte Carlo method. If you have not added users to HDFS (see Chapter 10, "Basic Hadoop Administration Pro- cedures"), run these tests as user hdfs. To run the pi example with 16 maps and 1,000,000 samples per map, enter the following command:
$ yarn jar $HADOOP_EXAMPLES/hadoop-mapreduce-examples.jar pi 16 1000000.

Using the Web GUI to Monitor Examples

This section provides an illustration of using the YARN ResourceManager web GUI to monitor and find information about YARN jobs. The Hadoop version 2 YARN Resource Manager web GUI differs significantly from the MapReduce web GUI found in Hadoop version 1. Figure 2.1 shows the main YARN web interface. The cluster metrics are displayed in the top row, while the running applications are displayed in the main table. A menu on the left provides navigation to the nodes table, various job categories (e.g., New, Accepted, Running, Finished, Failed), and the Capacity Sched- uler (covered in Chapter 10, "Basic Hadoop Administration Procedures"). This inter- face can be opened directly from the Ambari YARN service Quick Links menu or by directly entering http://hostname:8088 into a local web browser. For this example, the pi application is used. Note that the application can run quickly and may finish before you have fully explored the GUI. A longer-running application, such as terasort, may be helpful when exploring all the various links in the GUI.

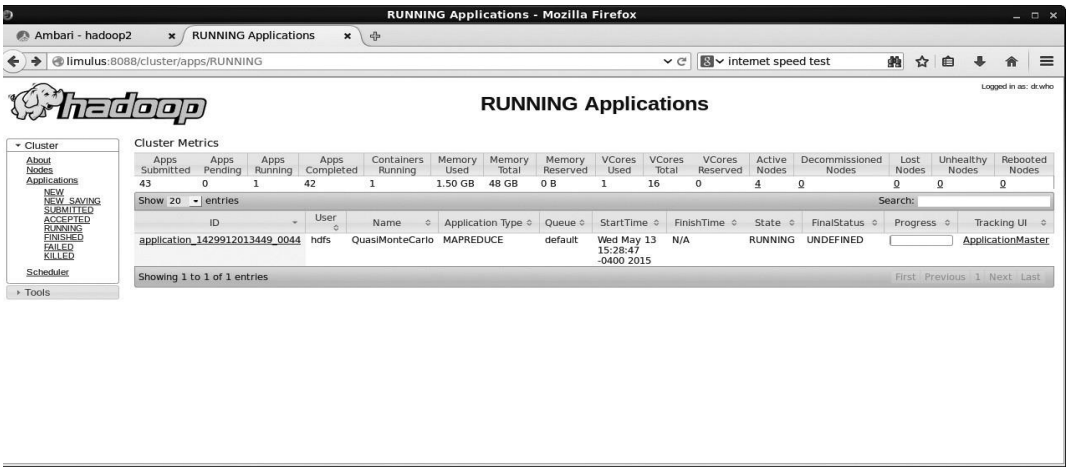


Figure 2.1 Hadoop RUNNING Applications web GUI for the pi example

For those readers who have used or read about Hadoop version 1, if you look at the Cluster Metrics table, you will see some new information. First, you will notice that the "Map/Reduce Task Capacity" has been replaced by the number of running containers. If YARN is running a MapReduce job, these containers can be used for both map and reduce tasks. Unlike in Hadoop version 1, the number of mappers and reducers is not fixed. There are also memory metrics and links to node status. If you click on the Nodes link (left menu under About), you can get a summary of the node activity and state. For example, Figure 2.2 is a snapshot of the node activity while the pi application is running. Notice the number of containers, which are used by the MapReduce framework as either mappers or reducers.

Going back to the main Applications/Running window (Figure 2.1), if you click on the application_14299… link, the Application status window in Figure 4.3 will appear. This window provides an application overview and metrics, including the clus- ter node on which the Application Master container is running.

Clicking the Application Master link next to "Tracking URL:" in Figure 2.3 leads to the window shown in Figure 4.4. Note that the link to the application's Application Master is also found in the last column on the main Running Applications screen shown in Figure 2.1.In the MapReduce Application window, you can see the details of the MapReduce application and the overall progress of mappers and reducers. Instead of containers, the MapReduce application now refers to maps and reducers. Clicking job_14299… brings up the window shown in Figure 2.5. This window displays more detail about the number of pending,

running, completed, and failed mappers and reducers, including the elapsed time since the job started.

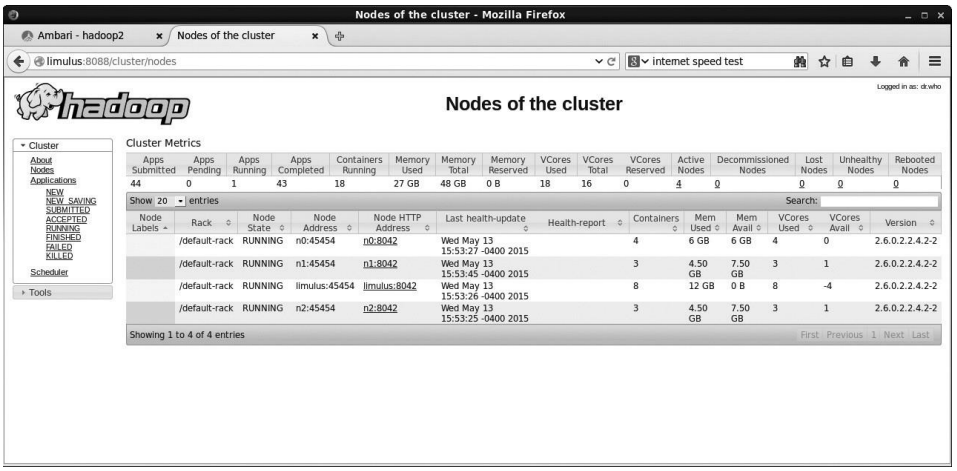


Figure 2.2 Hadoop YARN Resource Manager nodes status window

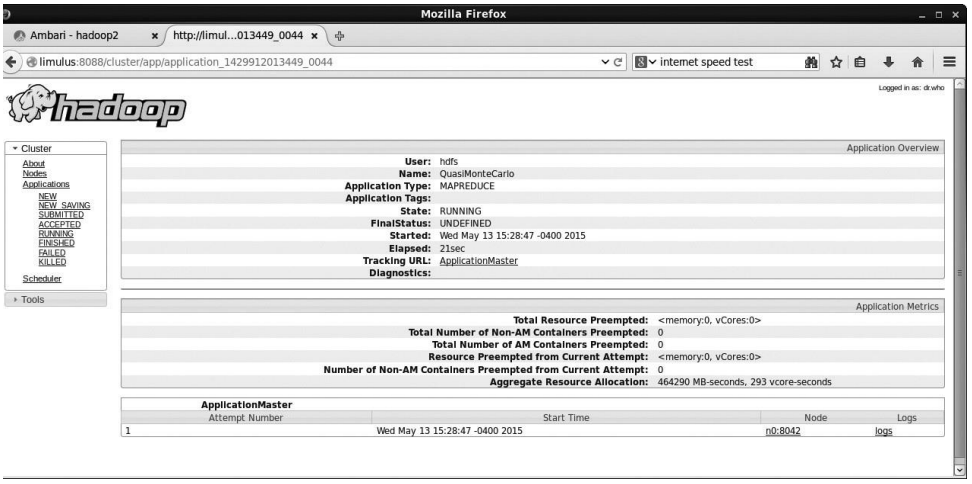


Figure 2.3 Hadoop YARN application status for the pi example

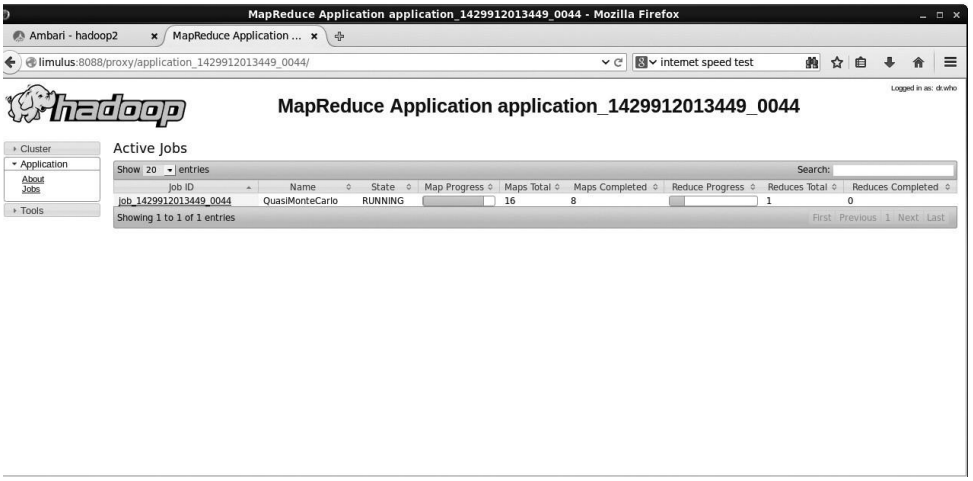


Figure 2.4 Hadoop YARN Application Master for Map Reduce application

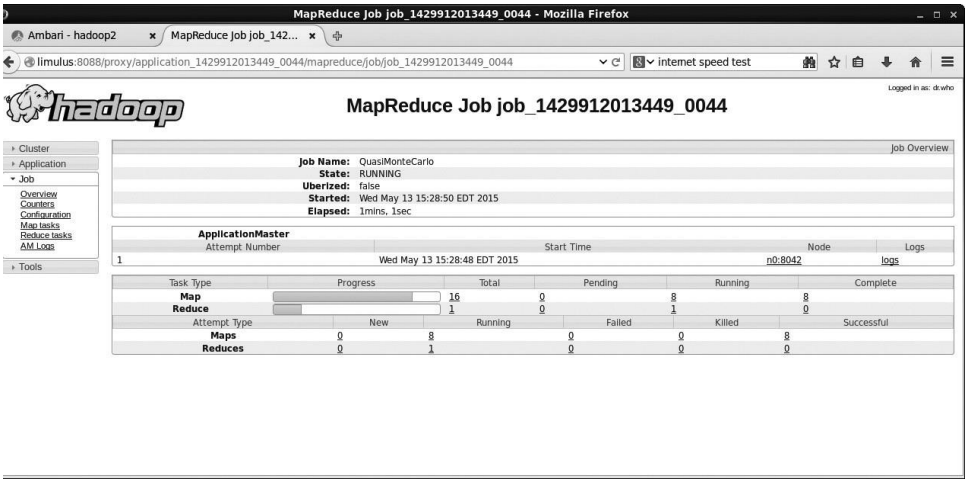


Figure 2.5 Hadoop YARN MapReduce job progress

The status of the job in Figure 2.5 will be updated as the job progresses (the win- dow needs to be refreshed manually). The ApplicationMaster collects and reports the progress of each mapper and reducer task. When the job is finished, the window is updated to that shown in Figure 2.6. It reports the overall run time and provides a breakdown of the timing of the key phases of the MapReduce job (map, shuffle, merge, reduce).

If you click the node used to run the Application Master (n0:8042 in Figure 2.6), the window in Figure 2.7 opens and provides a summary from the Node Manager on node n0. Again, the Node Manager tracks only containers; the actual tasks running in the containers are determined by the Application Master.

Going back to the job summary page (Figure 2.6), you can also examine the logs for the

Application Master by clicking the "logs" link. To find information about the mappers and reducers, click the numbers under the Failed, Killed, and Successful columns. In this example, there were 16 successful mappers and one successful reducer. All the numbers in these columns lead to more information about individual map or reduce process.

For instance, clicking the "16" under "Successful" in Figure 2 .6 displays the table of map tasks in Figure 2.8. The metrics for the Application Master container are displayed in table form. There is also a link to the log f ile for each process (in this case, a map process). Viewing the logs requires that the yarn.log.aggregation-enable variable in the yarn-site.xml file be set.

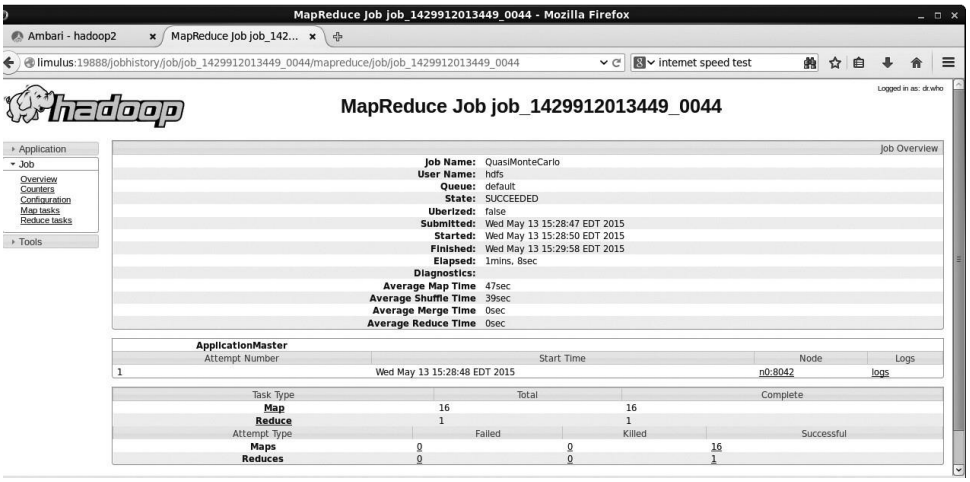


Figure 2.6 Hadoop YARN completed MapReduce job summary

Figure 2.7 Hadoop YARN Node Manager for n0 job summary

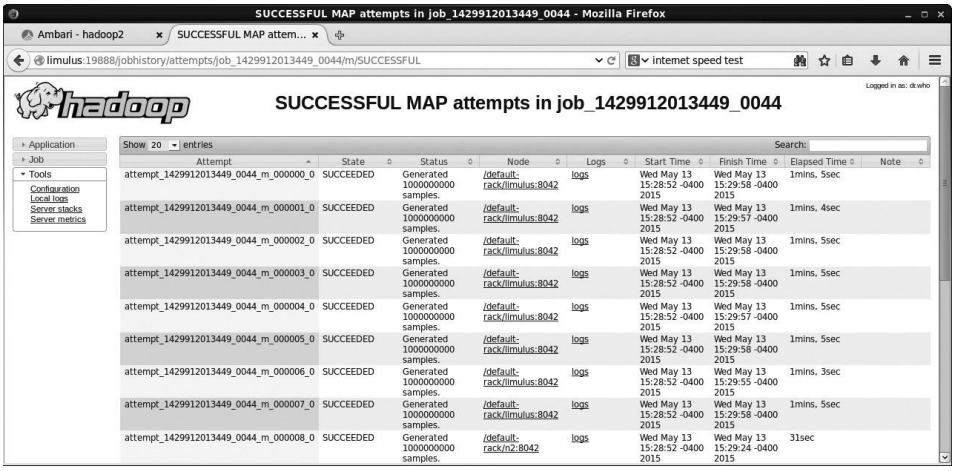


Figure 2.8 Hadoop YARN MapReduce logs available for browsing

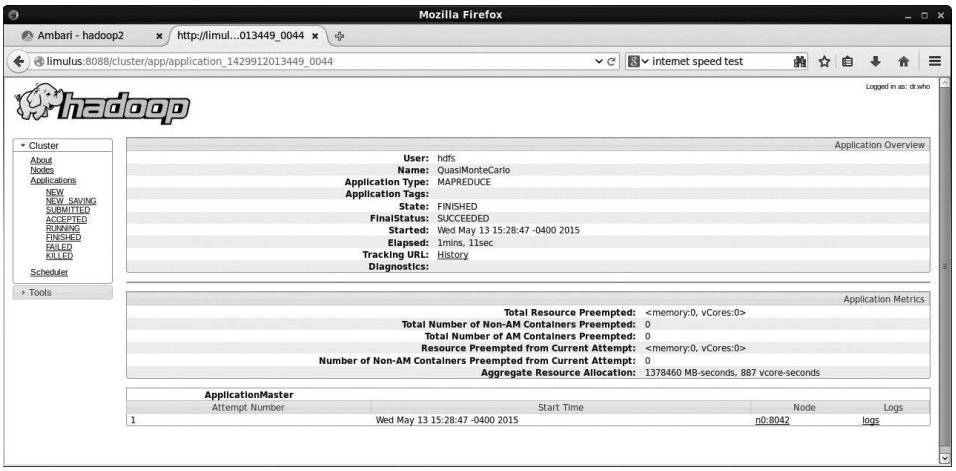


Figure 2.9 Hadoop YARN application summary page

If you return to the main cluster window (Figure 4.1), choose Applications/ Finished, and then select our application, you will see the summary page shown in Figure 2.9.

There are a few things to notice in the previous windows. First, because YARN manages applications, all information reported by the ResourceManager con-

cerns the resources provided and the application type (in this case, MAPREDUCE). In Figure 2.1 and Figure 2.4, the YARN ResourceManager refers to the pi example by its application-id (application_1429912013449_0044). YARN has no data about the actual application other than the fact that it is a MapReduce job. Data from the actual MapReduce job are provided by the MapReduce framework and referenced by a job- id (job_1429912013449_0044) in

Figure 4.6. Thus, two clearly different data streams are combined in the web GUI: YARN applications and MapReduce framework jobs. If the framework does not provide job information, then certain parts of the web GUI will not have anything to display.

Another interesting aspect of the previous windows is the dynamic nature of the mapper and reducer tasks. These tasks are executed as YARN containers, and their number will change as the application runs. Users may request specific numbers of mappers and reducers, but the ApplicationMaster uses them in a dynamic fashion. As mappers complete, the ApplicationMaster will return the containers to the Resource- Manager and request a smaller number of reducer containers. This feature provides for much better cluster utilization because mappers and reducers are dynamic—rather than fixed—resources.

Running Basic Hadoop Benchmarks

Many Hadoop benchmarks can provide insight into cluster performance. The best benchmarks are always those that ref lect real application performance. The two bench- marks discussed in this section, terasort and TestDFSIO, provide a good sense of how well your Hadoop installation is operating and can be compared with public data pub- lished for other Hadoop systems. The results, however, should not be taken as a single indicator for system-wide performance on all applications.

The following benchmarks are designed for full Hadoop cluster installations. These tests assume a multi-disk HDFS environment. Running these benchmarks in the Hor- tonworks Sandbox or in the pseudo-distributed single-node install from Chapter 2 is not recommended because all input and output (I/O) are done using a single system disk drive.

Running the Terasort Test

The terasort benchmark sorts a specified amount of randomly generated data. This benchmark provides combined testing of the HDFS and MapReduce layers of a Hadoop cluster. A full terasort benchmark run consists of the following three steps:

1. Generating the input data via teragen program.
2. Running the actual terasort benchmark on the input data.
3. Validating the sorted output data via the teravalidate program.

In general, each row is 100 bytes long; thus the total amount of data written is

100 times the number of rows specified as part of the benchmark (i.e., to write 100GB of data, use 1 billion rows). The input and output directories need to be specified in HDFS. The following sequence of commands will run the benchmark for 50GB of data as user hdfs. Make sure the /user/hdfs directory exists in HDFS before running the benchmarks.

1. Run tera gen to generate rows of random data to sort.

$ yarn jar $HADOOP_EXAMPLES/hadoop-mapreduce-examples.jar teragen 500000000 /user/hdfs/TeraGen-50GB

2. Run tera sort to sort the database.

$ yarn jar $HADOOP_EXAMPLES/hadoop-mapreduce-examples.jar tera sort /user/hdfs/TeraGen-50GB /user/hdfs/TeraSort-50GB

3. R tera validate to validate the sort.

$ yarn jar $HADOOP_EXAMPLES/hadoop-mapreduce-examples.jar tera validate /user/hdfs/TeraSort-50GB /user/hdfs/TeraValid-50GB

To report results, the time for the actual sort (terasort) is measured and the benchmark rate in megabytes/second (MB/s) is calculated. For best performance, the actual terasort benchmark should be run with a replication factor of 1. In addition, the default number of terasort reducer tasks is set to 1. Increasing the number of reducers often helps with benchmark performance. For example, the following com- mand will instruct terasort to use four reducer tasks:

$ yarn jar $HADOOP_EXAMPLES/hadoop-mapreduce-examples.jar terasort -Dmapred.reduce.tasks=4 /user/hdfs/TeraGen-50GB /user/hdfs/TeraSort-50GB

Also, do not forget to clean up the terasort data between runs (and after testing is finished). The following command will perform the cleanup for the previous example:

$ hdfs dfs -rm -r -skipTrash Tera*

Running the Test DFSIO Benchmark

Hadoop also includes an HDFS benchmark application called TestDFSIO. The TestDFSIO benchmark is a read and write test for HDFS. That is, it will write or read a number of files to and from HDFS and is designed in such a way that it will use one map task per file. The file size and number of files are specified as command-line arguments. Similar to the terasort benchmark, you should run this test as user hdfs.

Similar to terasort, TestDFSIO has several steps. In the following example,

16 files of size 1GB are specified. Note that the TestDFSIO benchmark is part of the hadoop-mapreduce-client-jobclient.jar. Other benchmarks are also available as part of this jar file. Running it with no arguments will yield a list. In addition to TestDFSIO, NNBench (load testing the NameNode) and MRBench (load testing the MapReduce frame- work) are commonly used Hadoop benchmarks. Nevertheless, TestDFSIO is perhaps the most widely reported of these benchmarks. The steps to run TestDFSIO are as follows:

1. Run TestDFSIO in write mode and create data.

```
$ yarn jar $HADOOP_EXAMPLES/hadoop-mapreduce-client-jobclient-tests.jar
TestDFSIO -write -nrFiles 16 -fileSize 1000
```

2. Run TestDFSIO in read mode.

```
$ yarn jar $HADOOP_EXAMPLES/hadoop-mapreduce-client-jobclient-tests.jar
TestDFSIO -read -nrFiles 16 -fileSize 1000
```

Managing Hadoop MapReduce Jobs

Hadoop MapReduce jobs can be managed using the mapred job command. The most important options for this command in terms of the examples and benchmarks are -list, -kill, and -status. In particular, if you need to kill one of the examples or benchmarks, you can use the mapred job –list command to find the job-id and then use mapred job –kill <job-id> to kill the job across the cluster. Map Reduce jobs can also be controlled at the application level with the yarn application command

chapter-3 ①

Hadoop MapReduce Framework

MapReduce programming model is conceptually simple. Based on two simple steps – applying mapping process and then reducing the result. It can be applied to many real-world problems.

The map Reduce model

The Apache Hadoop is associated with mapReduce computing. The map Reduce computation model provides a powerful tool for many applications and is more common than most user realize. There are two stages: A mapping stage and a reducing stage. In mapping stage, a mapping procedure is applied to input data. The mapping is usually some kind of filter or sorting process. Reducer usually shuffle and reduce the data.

The map Reduce model is inspired by the map and reduce functions commonly used in many functional programming languages.

1. Data flow is in one direction (map to reduce). It is possible to use the output of an reduce step as input to another map Reduce process.
2. As with functional programming, the input data are not changed. By applying the mapping and reduction function to the input data, raw data are produced.

3. Because there is no dependency on how the mapping and reducing functions are applied to the data, the mapper and reducer data flow can be implemented in any number of ways to provide better performance.

Map Reduce parallel Data flow

The programmer must provide a mapping function & a reducing function. Operationally Apache Hadoop parallel map Reduce data flow can be quite complex. parallel execution of map Reduce require other steps in addition to the mapper & reducer Process. The basic steps are as follows

1. **Input Split :** The input splits used by map Reduce are logical boundaries based on the input data.

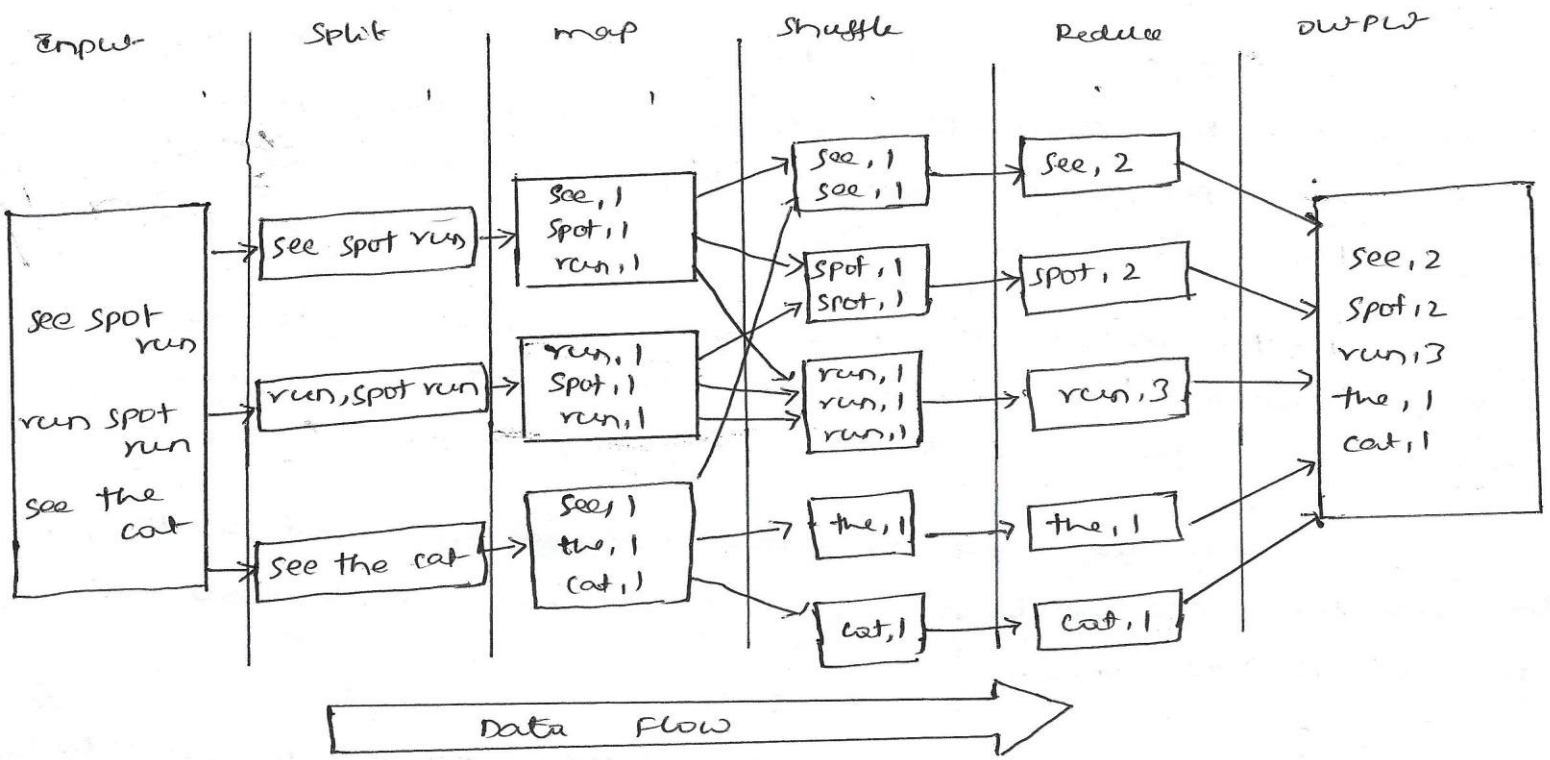
Eg:- The split size can be based on the number of records in a file or an actual size in bytes. splits are almost always smaller than the HDFS block size. The number of splits corresponds to the number of mapping process used in the map stage.

2. **map step :-** The mapping process is where the parallel nature of Hadoop comes into play. For larger amount of data, many mappers can be operating at the same time. The user provides the specific mapping process. map Reduce will try to execute the mapper on the machine where the block resides.

3. **Combiner step :-** It is possible to provide an optimization or pre reduction as part of the map stage where key-value pairs are combined prior to the next stage. The combiner stage is optional

4. **Shuffle Step :-** Before the parallel reduction stage can complete, all similar keys must be combined and counted by the same reducer process. Therefore, results of the map stage must be collected by key-value pair and shuffled to the same reducer process. (2)

5. **Reducer Step :-** The final step in actual reduction, In this stage actual data reduction is performed by Programmer's design. The reduce step is also optional.



Apache Hadoop parallel map Reduce data flow

Figure is an example of a simple Hadoop map Reduce data flow for a word count program. The map process counts the words in the split, and the reduce process

calculates the total for each word. The actual computation of the map and Reduce stages are upto the Programmer.

The input to the map Reduce application is the following file in HDFS with three lines of text. The goal is to count the number of times each word used.

See spot run
run spot run
see the cat

The first thing map Reduce will do is create the data splits. For simplicity each line will be one split. Since each split will require a map task. There are three mapper process that counts the number of words in split.

Next similar keys need to be collected and sent to a reducer process. The shuffle step require data movement and can be expensive in terms of processing time.

once the data have been collected and sorted by key, the reduction step can begin. The final step is to write the output to HDFS.

The combiner step enables some ~~Predefined~~ pre-reduction of the map output data.

(3)

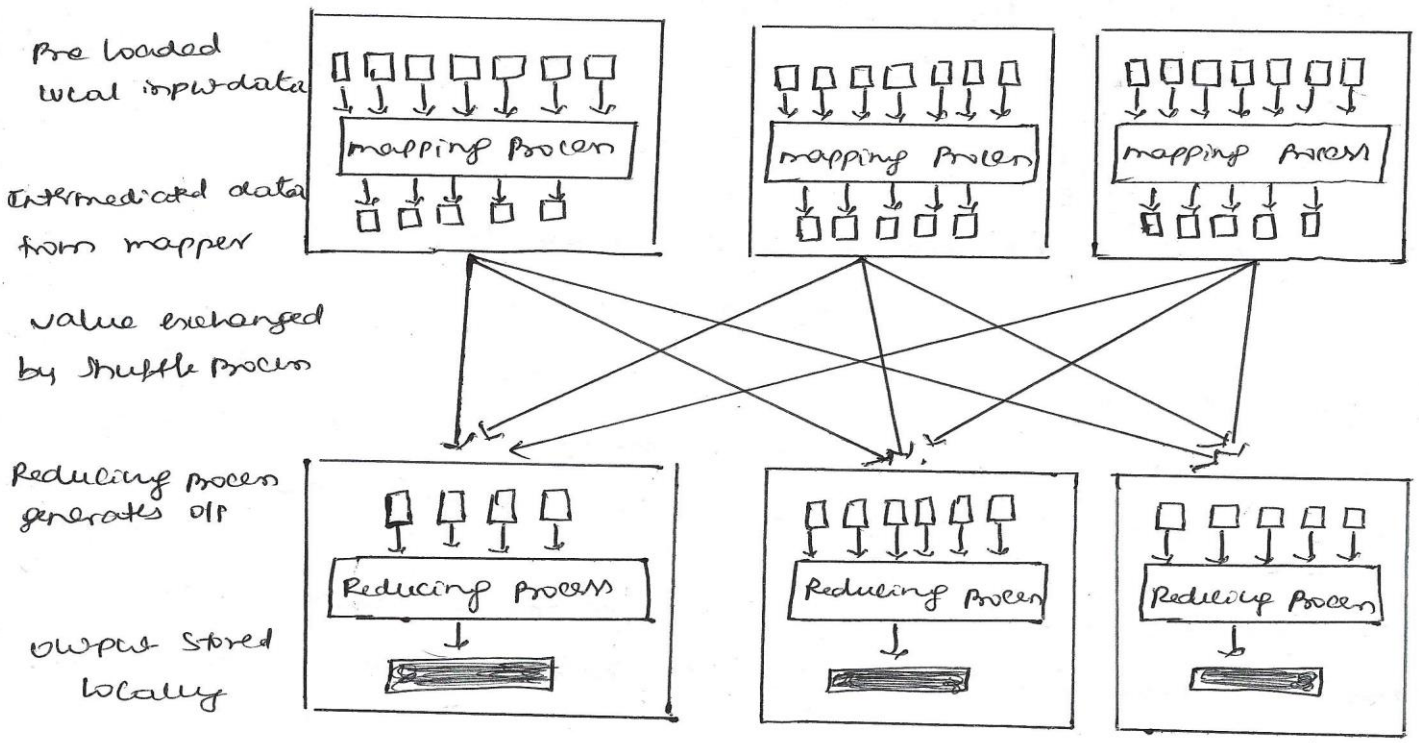


Fig: Process placement during Map Reduce

Figure shows a simple three node map reduce process. once the mapping is complete, the same nodes begin the reduce process. The shuffle stage makes sure the necessary data are sent to each mapper. The reducer can be set to start shuffling based on a threshold of percentage of mapper that have finished.

Fault Tolerance and Speculative execution

One of the interesting aspect of parallel map Reduce operation is the strict control of data flow throughout the execution of the program.

The design of map Reduce makes it possible to easily recover from the failure of one or many map process.

Eg. Should a server fail, the map task that were running on that machine could easily be restarted on another working server because there is no dependence on any other map task.

In similar fashion failed reducers can be restarted. However there may be additional work that has to be redone in such case.

Speculative execution

One of the challenge with many large clusters is the inability to predict or manage unexpected system bottlenecks or failures.

Eg.. congested n/w, slow disk controller, failure disk, high process load.

When one part of a map Reduce process runs slowly, it ultimately slow down everything else because the application can't complete until all processes are finished.

The secondary process finish first, the other first process are then terminated (or vice versa) This process is known as Speculative execution.

Hadoop map Reduce Hardware (4)

The capability of Hadoop map Reduce and HDFS to tolerate server - or even whole rack - failure can influence hardware design. The use of commodity server for Hadoop cluster has made low-cost, high-availability implementations of Hadoop possible for many data centers.

Apache Hadoop philosophy seems to assume server will always fail & takes steps to keep failure from stopping application progress on cluster.

chapter 4 Map Reduce Programming (1)

Hadoop provides a platform for Java-based map Reduce Programming. These applications run natively on most Hadoop installations. To offer more variability, a streaming interface is provided that enables almost any Programming language to take advantage of the Hadoop map Reduce Engine.

On addition, a pipe C++ interface is provided that can work directly with the map Reduce components.

compiling and Running the Hadoop word count example

To compile and run the program from command line perform the following steps.

1. make a local wordCount_classes directory

```
$mkdir wordcount_classes
```

2. Compile the wordCount.java Program using 'hadoop classpath'

```
$ javac -cp 'hadoop classpath' -d wordcount_classes
  wordcount.java
```

3. The jar file can be created using the following command

```
$ jar -cvf wordcount.jar -C wordcount_classes/
```

4. To run the example, create an input directory in HDFS & place a text file in a new directory.

```
$ hdfs dfs -mkdir war_and_peace_input
$ hdfs dfs -put war_and_peace.txt war_and_peace
                                   _input
```

5. Run the word count application using the following command

$ hadoop jar wordcount.jar WordCount war-and-peace-input

word count is a simple application that counts the number of occurance of each word in a given set.

The map reduce framework operates exclusively on key-value pairs, that is the framework views the input to the job as a set of key-value pairs and produces a set of key-value pairs of different types.

The map reduce job proceeds as follows

(input) $<K_1, V_1>$ → map → $<K_2, V_2>$ → combine → $<K_2, V_2>$ → reduce → $<K_3, V_3>$ (output).

Given two input files with contents 'Hello world Bye world' and 'Hello Hadoop Goodbye Hadoop' word count mapper will produce two maps.

<Hello, 1>
<world, 1>
<Bye, 1>
<world, 1>

<Hello, 1>
<Hadoop, 1>
<Goodbye, 1>
<Hadoop, 1>

```
/* word Count Java */

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

  public static class TokenizerMapper

       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);

    private Text word = new Text();

    public void map(Object key, Text value, Context context

                    ) throws IOException, InterruptedException {

      StringTokenizer itr = new StringTokenizer(value.toString());

      while (itr.hasMoreTokens()) {

        word.set(itr.nextToken());

        context.write(word, one);

      }

    }

  }
```

```
  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();
    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }
  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Using Streaming Interface (5)

The Apache Hadoop Streaming interface enables almost any program to use the map Reduce engine. The streams interface will work with any program that can read and write to stdin and stdout.

When working with in the Hadoop Streaming mode, only the mapper and the reducer are created by the user. This approach does have the advantage that the mapper and the reducer can be easily tested from the command line.

The operation of the mapper.py script can be observed by running the command as shown below

```
$ echo "foo foo quux labs foo bar quux | ./mapper.py
foo 1
foo 1
quux 1
labs 1
foo 1
bar 1
quux 1
```

Piping the result of the map into the sort command can create a simulated shuffle phase

```
$ echo " foo foo quux labs foo bar quux | ./mapper.py |
sort -k1,1
```

word count sets a mapper

```
job.setMapperClass(TokenizerMapper.class);
```

a combiner

```
job.setCombinerClass(IntSumReducer.class);
```

and a reducer

```
job.setReducerClass(IntSumReducer.class)
```

Hence, output of each map is passed through the local combiner for local aggregation and then sends the data on the final reducer. Thus each map above the combiner performs the following pre reduction.

<Bye, 1>
<Hello, 1>
<world, 2>

<Good bye, 1>
< Hadoop, 2>
<Hello, 1>

The reducer implementation via the reduce method, simply sums the value, which are the occurance counts for each key

The final output of the reducer is the following

<Bye, 1>
<Goodbye, 1>
< Hadoop, 2>
<Hello, 2>
<world, 2>

Bar 1
Foo 1
Foo 1
Foo 1
Labs 1
Qwerx 1
Qwerx 1

Finally, the full map reduce process can be simulated by adding the reducer.py script to the following command pipeline

$ echo "foo foo qwerx labs foo bar qwerx" | ./mapper.py | sort -k1,1 | ./reducer.py

Bar 1
Foo 2
Labs 1
Qwerx 2

```
#!/usr/bin/env python
"""mapper.py"""

import sys

# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    # increase counters
    for word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; the trivial word count is 1
        print '%s\t%s' % (word, 1)
```

```
#!/usr/bin/env python
"""reducer.py"""

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None
```

```
# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)

    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue

    # this IF-switch only works because Hadoop sorts map output
    # by key (here: word) before it is passed to the reducer
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # write result to STDOUT
            print '%s\t%s' % (current_word, current_count)
        current_count = count
        current_word = word

# do not forget to output the last word if needed!
if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```

Using the Pipes Interface

Pipes is a library that allows C++ source code to be used for mapper and reducer code. Applications that require high performance when crunching numbers may achieve better throughput if written in C++ and used through the Pipes interface.

Both key and value inputs to pipes programs are provided as STL strings. The programs must define an instance of a mapper and an instance of a reducer. A program to use with Pipes is defined by writing classes extending mapper and Reducer. Hadoop must then be informed as to which classes to use to run the job.

The Pipe framework on each machine assigned to your job will start an instance of C++ program. Therefore the executable must be placed in HDFS prior to use

Debugging map Reduce

(4)

The best advice for debugging parallel mapReduce application

The best approach is to make sure application runs on a simpler system with smaller data sets. Errors on these systems are much easier to locate and track.

If applications can run successfully on a single system with a subset of real data, then running in parallel should be simple task because the map Reduce algorithm is transparently scalable.

Listing, Killing and Job Status

Jobs can be managed using the mapred job command. The important options are -list, -kill, and -status. In addition the yarn application command can be used to control all applications running on cluster.

Hadoop log management

The mapReduce logs provide a comprehensive listing of both mapper and reducer. The actual log output consists of three files - stdout, stderr & syslog for the application.

There are two modes for log storage

(i) log aggregation : logs are aggregated in HDFS and can be displayed in the YARN Resource manager user interface.

If log aggregation is not enabled, the logs will be placed locally on the cluster nodes on which mapper or reducer ran. Log aggregation is highly recommended.

Enabling YARN log aggregation

If Apache Hadoop was installed from the official Apache Hadoop source the following settings will ensure log aggregation is turned on for the system.

To manually enable log aggregation, follow these steps

1. As the HDFS superuser administrator, create a following directory

```
$ hdfs dfs -mkdir -p /var/log/hadoop-yarn
$ hdfs dfs -chown -R yarn:hadoop /var/log/hadoop-yarn
$ hdfs dfs -chmod -R g+rw /var/log/hadoop-yarn
```

2. Add the properties in the yarn-site.xml and restart all YARN services on all the nodes (the Resource manager, Node manager, and Job History server).

web interface log view (5)

The most convenient way to view logs is to use the YARN resource manager web user interface. Each task has a link to the logs for that task. If log aggregation is enabled, clicking on the log link will show a window.

The contents of stdout, stderr, syslog are displayed on a single page. If log aggregation is not enabled, a message stating that the logs are not available will be displayed.

command-line log viewing

Map Reduce logs can also be viewed from a command line. The yarn log command enables the logs to be easily viewed together without having to hunt for individual log files on the cluster nodes. Log aggregation is required for use.

The options of yarn log are as follows

$yarn logs

general options

-appOwner	<Application owner>	assume to be current user
-containerID	<container ID>	must be specified if node address is specified
-nodeAddress	<Node Address>	must be specified if container id is specified

Module 2
1. Essential Hadoop Tools

In This Chapter:

■ The Pig scripting tool is introduced as a way to quickly examine data both locally and on a Hadoop cluster.
■ The Hive SQL-like query tool is explained using two examples.
■ The Sqoop RDBMS tool is used to import and export data from MySQL to/from HDFS.
■ The Flume streaming data transport utility is configured to capture weblog data into HDFS.
■ The Oozie workflow manager is used to run basic and complex Hadoop workflows.
■ The distributed HBase database is used to store and access data on a Hadoop cluster.

USING APACHE PIG

Apache Pig is a high-level language that enables programmers to write complex MapReduce transformations using a simple scripting language. Pig Latin (the actual language) defines a set of transformations on a data set such as aggregate, join, and sort.

Apache Pig has several usage modes.

- The first is a local mode in which all processing is done on the local machine.
- The non-local (cluster) modes are MapReduce and Tez. These modes execute the job on the cluster using either the MapReduce engine or the optimized Tez engine.

There are also interactive and batch modes available; they enable Pig applications to be developed locally in interactive modes, using small amounts of data, and then run at scale on the cluster in a production mode. The modes are summarized in Table 7.1.

	Local Mode	Tez Local Mode	MapReduce Mode	Tez Mode
Interactive Mode	Yes	Experimental	Yes	Yes
Batch Mode	Yes	Experimental	Yes	Yes

Table 7.1 **Apache Pig Usage Modes**

Pig Example Walk-Through

In this simple example, Pig is used The following example assumes the user is hdfs, but any valid user with access to HDFS can run the example.

To begin the example, copy the passwd file to a working directory for local Pig operation:

```
$ cp /etc/passwd .
```

Next, copy the data file into HDFS for Hadoop MapReduce operation:

```
$ hdfs dfs -put passwd passwd
```

You can confirm the file is in HDFS by entering the following command:

```
hdfs dfs -ls passwd
-rw-r--r--   2 hdfs hdfs       2526 2015-03-17 11:08 passwd
```

In the following example of local Pig operation, all processing is done on the local machine (Hadoop is not used). First, the interactive command line is started:

```
$ pig -x local
```

If Pig starts correctly, you will see a grunt> prompt. Next, enter the following commands to load the passwd file and then grab the user name and dump it to the terminal. Note that Pig commands must end with a semicolon (;).

```
grunt> A = load 'passwd' using PigStorage(':');
grunt> B = foreach A generate $0 as id;
grunt> dump B;
```

The processing will start and a list of user names will be printed to the screen. To exit the interactive session, enter the command quit.

```
$ grunt> quit
```

To use Hadoop MapReduce, start Pig as follows (or just enter pig):

```
$ pig -x mapreduce
```

The same sequence of commands can be entered at the grunt> prompt. You may wish to change the $0 argument to pull out other items in the passwd file. Also, because we are running this application under Hadoop, make sure the file is placed in HDFS.

If you are using the Hortonworks HDP distribution with tez installed, the tez engine can be used as follows:

```
$ pig -x tez
```

Pig can also be run from a script. This script, which is repeated here, is designed to do the same things as the interactive version:

```
/* id.pig */
A = load 'passwd' using PigStorage(':'); -- load the passwd file
B = foreach A generate $0 as id; -- extract the user IDs
dump B;
store B into 'id.out'; -- write the results to a directory name id.out
```

Comments are delineated by /* */ and -- at the end of a line. First, ensure that the id.out directory is not in your local directory, and then start Pig with the script on the command line:

```
$ /bin/rm -r id.out/
$ pig -x local id.pig
```

If the script worked correctly, you should see at least one data file with the results and a zero-length file with the name _SUCCESS. To run the MapReduce version, use the same procedure; the only difference is that now all reading and writing takes place in HDFS.

```
$ hdfs dfs -rm -r id.out
$ pig id.pig
```

USING APACHE HIVE

Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, ad hoc queries, and the analysis of large data sets using a SQL-like language called HiveQL. Hive offers the following features:

- Tools to enable easy data extraction, transformation, and loading (ETL)
- A mechanism to impose structure on a variety of data formats
- Access to files stored either directly in HDFS or in other data storage systems such as HBase
- Query execution via MapReduce and Tez (optimized MapReduce)

Hive Example Walk-Through

To start Hive, simply enter the hive command. If Hive starts correctly, you should get a hive> prompt.

```
$ hive
(some messages may show up here)
hive>
```

As a simple test, create and drop a table. Note that Hive commands must end with a semicolon (;).

```
hive> CREATE TABLE pokes (foo INT, bar STRING);
OK
Time taken: 1.705 seconds
hive> SHOW TABLES;
OK
pokes
Time taken: 0.174 seconds, Fetched: 1 row(s)
hive> DROP TABLE pokes;
OK
Time taken: 4.038 seconds
```

A more detailed example can be developed using a web server log file to summarize message types. First, create a table using the following command:

```
hive> CREATE TABLE logs(t1 string, t2 string, t3 string, t4 string, t5 string, t6 string, t7 string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ' ';
OK
Time taken: 0.129 seconds
```

Next, load the data—in this case, from the sample.log file. Note that the file is found in the local directory and not in HDFS.

```
hive> LOAD DATA LOCAL INPATH 'sample.log' OVERWRITE INTO TABLE logs;
Loading data to table default.logs
Table default.logs stats: [numFiles=1, numRows=0, totalSize=99271, rawDataSize=0]
OK
Time taken: 0.953 seconds
```

Finally, apply the select step to the file. Note that this invokes a Hadoop MapReduce operation. The results appear at the end of the output (e.g., totals for the message types DEBUG, ERROR, and so on).

```
hive> SELECT t4 AS sev, COUNT(*) AS cnt FROM logs WHERE t4 LIKE '[%' GROUP BY t4;
Query ID = hdfs_20150327130000_d1e1a265-a5d7-4ed8-b785-2c6569791368
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
```

```
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1427397392757_0001, Tracking URL = http://norbert:8088/proxy/
application_1427397392757_0001/
Kill Command = /opt/hadoop-2.6.0/bin/hadoop job -kill job_1427397392757_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2015-03-27 13:00:17,399 Stage-1 map = 0%, reduce = 0%
2015-03-27 13:00:26,100 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.14 sec
2015-03-27 13:00:34,979 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 4.07 sec
MapReduce Total cumulative CPU time: 4 seconds 70 msec
Ended Job = job_1427397392757_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 4.07 sec HDFS Read: 106384
HDFS Write: 63 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 70 msec
OK
[DEBUG] 434
[ERROR] 3
[FATAL] 1
[INFO]  96
[TRACE] 816
[WARN] 4
Time taken: 32.624 seconds, Fetched: 6 row(s)
```

To exit Hive, simply type exit;

```
hive> exit;
```

A More Advanced Hive Example

In this example, 100,000 records will be transformed from userid, movieid, rating, unixtime to userid, movieid, rating, and weekday using Apache Hive and a Python program (i.e., the UNIX time notation will be transformed to the day of the week). The first step is to download and extract the data:

```
$ wget http://files.grouplens.org/datasets/movielens/ml-100k.zip
$ unzip ml-100k.zip
$ cd ml-100k
```

Before we use Hive, we will create a short Python program called weekday_mapper.py with following contents:

```
import sys
import datetime

for line in sys.stdin:
  line = line.strip()
  userid, movieid, rating, unixtime = line.split('\t')
  weekday = datetime.datetime.fromtimestamp(float(unixtime)).isoweekday()
  print '\t'.join([userid, movieid, rating, str(weekday)])LOAD DATA LOCAL INPATH './u.data'
OVERWRITE INTO TABLE u_data;
```

Next, start Hive and create the data table (u_data) by entering the following at the hive> prompt:

```
CREATE TABLE u_data (
  userid INT,
  movieid INT,
  rating INT,
  unixtime STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE;
```

Load the movie data into the table with the following command:

```
hive> LOAD DATA LOCAL INPATH './u.data' OVERWRITE INTO TABLE u_data;
```

The number of rows in the table can be reported by entering the following command:

```
hive > SELECT COUNT(*) FROM u_data;
```

This command will start a single MapReduce job and should finish with the following lines:

```
...
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 2.26 sec HDFS Read: 1979380
HDFS Write: 7 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 260 msec
OK
100000
Time taken: 28.366 seconds, Fetched: 1 row(s)
```

Now that the table data are loaded, use the following command to make the new table (u_data_new):

```
hive> CREATE TABLE u_data_new (
  userid INT,
  movieid INT,
  rating INT,
  weekday INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t';
```

The next command adds the weekday_mapper.py to Hive resources:

```
hive> add FILE weekday_mapper.py;
```

Once weekday_mapper.py is successfully loaded, we can enter the transformation query:

```
hive> INSERT OVERWRITE TABLE u_data_new
SELECT
  TRANSFORM (userid, movieid, rating, unixtime)
  USING 'python weekday_mapper.py'
  AS (userid, movieid, rating, weekday)
FROM u_data;
```

If the transformation was successful, the following final portion of the output should be displayed:

```
...
Table default.u_data_new stats: [numFiles=1, numRows=100000, totalSize=1179173,
rawDataSize=1079173]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 3.44 sec HDFS Read: 1979380 HDFS Write:
1179256 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 440 msec
```

```
OK
Time taken: 24.06 seconds
```

The final query will sort and group the reviews by weekday:

```
hive> SELECT weekday, COUNT(*) FROM u_data_new GROUP BY weekday;
```

Final output for the review counts by weekday should look like the following:

```
...
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 2.39 sec HDFS Read: 1179386
HDFS Write: 56 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 390 msec
OK
1	13278
2	14816
3	15426
4	13774
5	17964
6	12318
7	12424
Time taken: 22.645 seconds, Fetched: 7 row(s)
```

As shown previously, you can remove the tables used in this example with the DROP TABLE command. In this case, we are also using the -e command-line option. Note that queries can be loaded from files using the -f option as well.

```
$ hive -e 'drop table u_data_new'
$ hive -e 'drop table u_data'
```

USING APACHE SQOOP TO ACQUIRE RELATIONAL DATA

Sqoop is a tool designed to transfer data between Hadoop and relational databases. You can use Sqoop to import data from a relational database management system (RDBMS) into the Hadoop Distributed File System (HDFS), transform the data in Hadoop, and then export the data back into an RDBMS.

Sqoop can be used with any Java Database Connectivity (JDBC)–compliant database and has been tested on Microsoft SQL Server, PostgresSQL, MySQL, and Oracle.

Apache Sqoop Import and Export Methods

Figure 7.1 describes the Sqoop data import (to HDFS) process. The data import is done in two steps. In the first step, shown in the figure, Sqoop examines the database to gather the necessary metadata for the data to be imported. The second step is a map-only (no reduce step) Hadoop job that Sqoop submits to the cluster. This job does the actual data transfer using the metadata captured in the previous step. Note that each node doing the import must have access to the database.

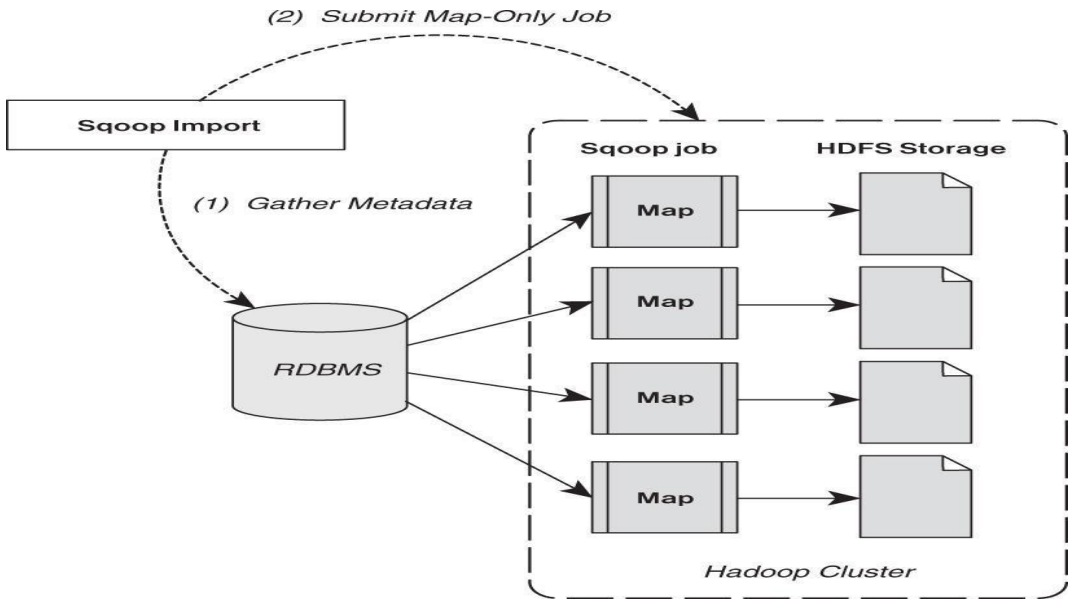


Figure 7.1 Two-step Apache Sqoop data import method (Adapted from Apache Sqoop Documentation)

The imported data are saved in an HDFS directory. Sqoop will use the database name for the directory, or the user can specify any alternative directory where the files should be populated. By default, these files contain comma-delimited fields, with new lines separating different records. You can easily override the format in which data are copied over by explicitly specifying the field separator and record terminator characters. Once placed in HDFS, the data are ready for processing.

Data export from the cluster works in a similar fashion. The export is done in two steps, as shown in Figure 7.2. As in the import process, the first step is to examine the database for metadata. The export step again uses a map-only Hadoop job to write the data to the database. Sqoop divides the input data set into splits, then uses individual map tasks to push the splits to the database. Again, this process assumes the map tasks have access to the database.

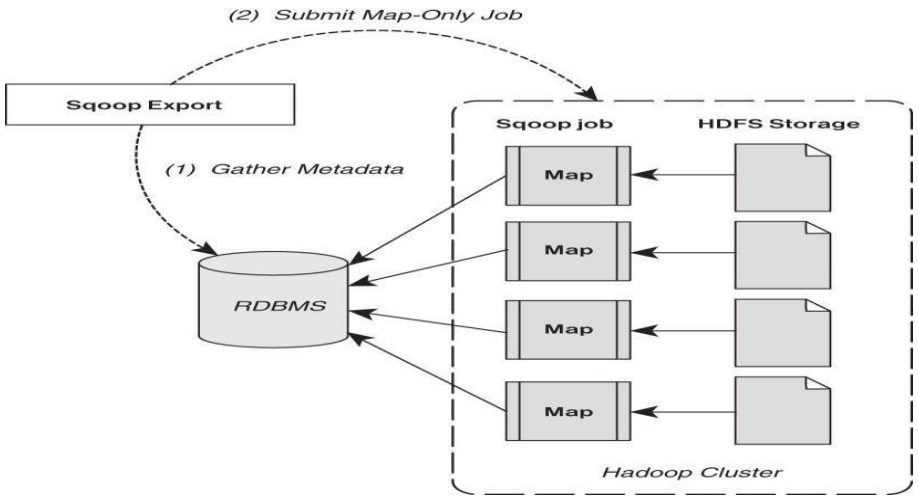


Figure 7.2 Two-step Sqoop data export method (Adapted from Apache Sqoop Documentation)

Apache Sqoop Version Changes

Sqoop Version 1 uses specialized connectors to access external systems. These connectors are often optimized for various RDBMSs or for systems that do not support JDBC. Connectors are plug-in components based on Sqoop's extension framework and can be added to any existing Sqoop installation. Once a connector is installed, Sqoop can use it to efficiently transfer data between Hadoop and the external store supported by the connector. By default, Sqoop version 1 includes connectors for popular databases such as MySQL, PostgreSQL, Oracle, SQL Server, and DB2. It also supports direct transfer to and from the RDBMS to HBase or Hive.

In contrast, to streamline the Sqoop input methods, Sqoop version 2 no longer supports specialized connectors or direct import into HBase or Hive. All imports and exports are done through the JDBC interface. Table 7.2 summarizes the changes from version 1 to version 2. Due to these changes, any new development should be done with Sqoop version 2.

Feature	Sqoop Version 1	Sqoop Version 2
Connectors for all major RDBMSs	Supported.	Not supported. Use the generic JDBC connector.
Kerberos security integration	Supported.	Not supported.
Data transfer from RDBMS to Hive or HBase	Supported.	Not supported. First import data from RDBMS into HDFS, then load data into Hive or HBase manually.
Data transfer from Hive or HBase to RDBMS	Not supported. First export data from Hive or HBase into HDFS, and then use Sqoop for export.	Not supported. First export data from Hive or HBase into HDFS, then use Sqoop for export.

Table 7.2 **Apache Sqoop Version Comparison**

Sqoop Example Walk-Through

The following simple example illustrates use of Sqoop

Step 1: Load Sample MySQL Database

```
$ wget http://downloads.mysql.com/docs/world_innodb.sql.gz
$ gunzip world_innodb.sql.gz
```

Next, log into MySQL (assumes you have privileges to create a database) and import the desired database by following these steps:

```
$ mysql -u root -p
mysql> CREATE DATABASE world;
mysql> USE world;
mysql> SOURCE world_innodb.sql;
mysql> SHOW TABLES;
+-----------------+
| Tables_in_world |
+-----------------+
| City            |
| Country         |
| CountryLanguage |
+-----------------+
3 rows in set (0.01 sec)
```

The following MySQL command will let you see the table details.

Step 2: Add Sqoop User Permissions for the Local Machine and Cluster

In MySQL, add the following privileges for user sqoop to MySQL. Note that you must use both the local host name and the cluster subnet for Sqoop to work properly. Also, for the purposes of this example, the sqoop password is sqoop.

```
mysql> GRANT ALL PRIVILEGES ON world.* To 'sqoop'@'limulus' IDENTIFIED BY 'sqoop';
mysql> GRANT ALL PRIVILEGES ON world.* To 'sqoop'@'10.0.0.%' IDENTIFIED BY 'sqoop';
mysql> quit
```

Next, log in as sqoop to test the permissions:

```
$ mysql -u sqoop -p
mysql> USE world;
mysql> SHOW TABLES;
+-----------------+
| Tables_in_world |
+-----------------+
| City            |
| Country         |
| CountryLanguage |
+-----------------+
3 rows in set (0.01 sec)

mysql> quit
```

Step 3: Import Data Using Sqoop

As a test, we can use Sqoop to list databases in MySQL. The results appear after the warnings at the end of the output. Note the use of local host name (limulus) in the JDBC statement.

```
$ sqoop list-databases --connect jdbc:mysql://limulus/world --username sqoop --password sqoop
  Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
  Please set $ACCUMULO_HOME to the root of your Accumulo installation.
  14/08/18 14:38:55 INFO sqoop.Sqoop: Running Sqoop version: 1.4.4.2.1.2.1-471
  14/08/18 14:38:55 WARN tool.BaseSqoopTool: Setting your password on the
command-line is insecure. Consider using -P instead.
  14/08/18 14:38:55 INFO manager.MySQLManager: Preparing to use a MySQL streaming
resultset.
  information_schema
```

```
test
world
```

In a similar fashion, you can use Sqoop to connect to MySQL and list the tables in the world database:

```
sqoop list-tables --connect jdbc:mysql://limulus/world --username sqoop --password sqoop
...
14/08/18 14:39:43 INFO sqoop.Sqoop: Running Sqoop version: 1.4.4.2.1.2.1-471
14/08/18 14:39:43 WARN tool.BaseSqoopTool: Setting your password on the
command-line is insecure. Consider using -P instead.
14/08/18 14:39:43 INFO manager.MySQLManager: Preparing to use a MySQL streaming
resultset.
City
Country
CountryLanguage
```

To import data, we need to make a directory in HDFS:

```
$ hdfs dfs -mkdir sqoop-mysql-import
```

The following command imports the Country table into HDFS. The option -table signifies the table to import, --target-dir is the directory created previously, and -m 1 tells Sqoop to use one map task to import the data.

```
$ sqoop import --connect jdbc:mysql://limulus/world --username sqoop --password sqoop --table
Country -m 1 --target-dir /user/hdfs/sqoop-mysql-import/country
...
14/08/18 16:47:15 INFO mapreduce.ImportJobBase: Transferred 30.752 KB in
12.7348 seconds
(2.4148 KB/sec)
14/08/18 16:47:15 INFO mapreduce.ImportJobBase: Retrieved 239 records.
```

The import can be confirmed by examining HDFS:

```
$ hdfs dfs -ls sqoop-mysql-import/country
Found 2 items
-rw-r--r--   2 hdfs hdfs          0 2014-08-18 16:47 sqoop-mysql-import/
world/_SUCCESS
-rw-r--r--   2 hdfs hdfs      31490 2014-08-18 16:47 sqoop-mysql-import/world/
part-m-00000
```

The file can be viewed using the hdfs dfs -cat command:

```
$ hdfs dfs -cat sqoop-mysql-import/country/part-m-00000
ABW,Aruba,North America,Caribbean,193.0,null,103000,78.4,828.0,793.0,Aruba,
Nonmetropolitan
Territory of The Netherlands,Beatrix,129,AW
...
ZWE,Zimbabwe,Africa,Eastern Africa,390757.0,1980,11669000,37.8,5951.0,8670.0,
Zimbabwe,
Republic,Robert G. Mugabe,4068,ZW
```

To make the Sqoop command more convenient, you can create an options file and use it on the command line. Such a file enables you to avoid having to rewrite the same options. For

example, a file called world-options.txt with the following contents will include the import command, --connect, --username, and --password options:

```
import
--connect
jdbc:mysql://limulus/world
--username
sqoop
--password
sqoop
```

The same import command can be performed with the following shorter line:

$ **sqoop --options-file world-options.txt --table City -m 1 --target-dir /user/hdfs/sqoop-mysql-import/city**

It is also possible to include an SQL Query in the import step. For example, suppose we want just cities in Canada:

SELECT ID,Name from City WHERE CountryCode='CAN'

In such a case, we can include the --query option in the Sqoop import request. The --query option also needs a variable called $CONDITIONS, which will be explained next. In the following query example, a single mapper task is designated with the -m 1 option:

sqoop --options-file world-options.txt -m 1 --target-dir /user/hdfs/sqoop-mysql-import/canada-city --query "SELECT ID,Name from City WHERE CountryCode='CAN' AND \$CONDITIONS"

Inspecting the results confirms that only cities from Canada have been imported:

```
$ hdfs dfs -cat sqoop-mysql-import/canada-city/part-m-00000
1810,MontrÄal
1811,Calgary
1812,Toronto
...
1856,Sudbury
1857,Kelowna
1858,Barrie
```

Since there was only one mapper process, only one copy of the query needed to be run on the database. The results are also reported in a single file (part-m-0000).

Multiple mappers can be used to process the query if the --split-by option is used. The split-by option is used to parallelize the SQL query. Each parallel task runs a subset of the main query, with the results of each sub-query being partitioned by bounding conditions inferred by Sqoop. Your query must include the token $CONDITIONS that each Sqoop process will replace with a unique condition expression based on the --split-by option. Note that $CONDITIONS is not an environment variable. Although Sqoop will try to create

balanced sub-queries based on the range of your primary key, it may be necessary to split on another column if your primary key is not uniformly distributed.

The following example illustrates the use of the --split-by option. First, remove the results of the previous query:

```
$ hdfs dfs -rm -r -skipTrash sqoop-mysql-import/canada-city
```

Next, run the query using four mappers (-m 4), where we split by the ID number (--split-by ID):

```
sqoop --options-file world-options.txt -m 4 --target-dir /user/hdfs/sqoop-mysql-import/canada-city --
query "SELECT ID,Name from City WHERE CountryCode='CAN' AND \$CONDITIONS" --split-by
ID
```

If we look at the number of results files, we find four files corresponding to the four mappers we requested in the command:

```
$ hdfs dfs -ls sqoop-mysql-import/canada-city
Found 5 items
-rw-r--r--   2 hdfs hdfs       0 2014-08-18 21:31 sqoop-mysql-import/
canada-city/_SUCCESS
-rw-r--r--   2 hdfs hdfs     175 2014-08-18 21:31 sqoop-mysql-import/canada-city/
part-m-00000
-rw-r--r--   2 hdfs hdfs     153 2014-08-18 21:31 sqoop-mysql-import/canada-city/
part-m-00001
-rw-r--r--   2 hdfs hdfs     186 2014-08-18 21:31 sqoop-mysql-import/canada-city/
part-m-00002
-rw-r--r--   2 hdfs hdfs     182 2014-08-18 21:31 sqoop-mysql-import/canada-city/
part-m-00003
```

Step 4: Export Data from HDFS to MySQL

Sqoop can also be used to export data from HDFS. The first step is to create tables for exported data. There are actually two tables needed for each exported table. The first table holds the exported data (CityExport), and the second is used for staging the exported data (CityExportStaging). Enter the following MySQL commands to create these tables:

```
mysql> CREATE TABLE 'CityExport' (
       'ID' int(11) NOT NULL AUTO_INCREMENT,
       'Name' char(35) NOT NULL DEFAULT '',
       'CountryCode' char(3) NOT NULL DEFAULT '',
       'District' char(20) NOT NULL DEFAULT '',
       'Population' int(11) NOT NULL DEFAULT '0',
       PRIMARY KEY ('ID'));
mysql> CREATE TABLE 'CityExportStaging' (
       'ID' int(11) NOT NULL AUTO_INCREMENT,
       'Name' char(35) NOT NULL DEFAULT '',
       'CountryCode' char(3) NOT NULL DEFAULT '',
       'District' char(20) NOT NULL DEFAULT '',
       'Population' int(11) NOT NULL DEFAULT '0',
       PRIMARY KEY ('ID'));
```

Next, create a cities-export-options.txt file similar to the world-options.txt created previously, but use the export command instead of the import command.

The following command will export the cities data we previously imported back into MySQL:

sqoop --options-file cities-export-options.txt --table CityExport --staging-table CityExportStaging --clear-staging-table -m 4 --export-dir /user/hdfs/sqoop-mysql-import/city

Finally, to make sure everything worked correctly, check the table in MySQL to see if the cities are in the table:

```
$ mysql> select * from CityExport limit 10;
+----+----------------+-------------+---------------+------------+
| ID | Name           | CountryCode | District      | Population |
+----+----------------+-------------+---------------+------------+
|  1 | Kabul          | AFG         | Kabol         | 1780000 |
|  2 | Qandahar       | AFG         | Qandahar      |     237500 |
|  3 | Herat          | AFG         | Herat         |     186800 |
|  4 | Mazar-e-Sharif | AFG         | Balkh         |     127800 |
|  5 | Amsterdam      | NLD         | Noord-Holland |     731200 |
|  6 | Rotterdam      | NLD         | Zuid-Holland  |     593321 |
|  7 | Haag           | NLD         | Zuid-Holland  |     440900 |
|  8 | Utrecht        | NLD         | Utrecht       |     234323 |
|  9 | Eindhoven      | NLD         | Noord-Brabant |     201843 |
| 10 | Tilburg        | NLD         | Noord-Brabant |     193238 |
+----+----------------+-------------+---------------+------------+
10 rows in set (0.00 sec)
```

Some Handy Cleanup Commands

If you are not especially familiar with MySQL, the following commands may be helpful to clean up the examples. To remove the table in MySQL, enter the following command:

mysql> **drop table 'CityExportStaging';**

To remove the data in a table, enter this command:

mysql> **delete from CityExportStaging;**

To clean up imported files, enter this command:

$ **hdfs dfs -rm -r -skipTrash sqoop-mysql-import/{country,city, canada-city}**

USING APACHE FLUME TO ACQUIRE DATA STREAMS

Apache Flume is an independent agent designed to collect, transport, and store data into HDFS. Often data transport involves a number of Flume agents that may traverse a series of machines and locations. Flume is often used for log files, social media-generated data, email messages, and just about any continuous data source. As shown in Figure 7.3, a Flume agent is composed of three components.

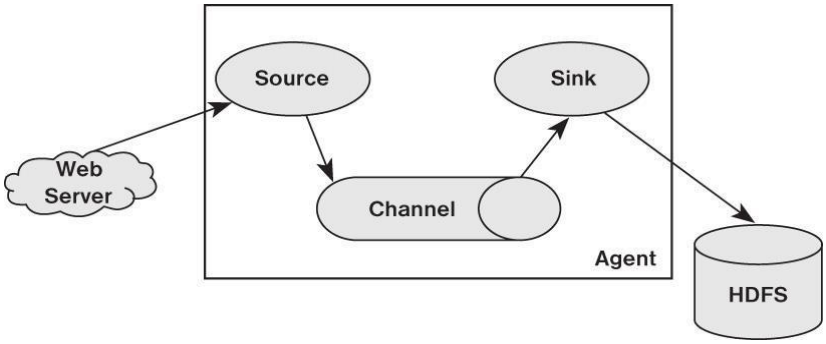


Figure 7.3 Flume agent with source, channel, and sink (Adapted from Apache Flume Documentation)

■ *Source.* The source component receives data and sends it to a channel. It can send the data to more than one channel. The input data can be from a real-time source (e.g., weblog) or another Flume agent.

■ *Channel.* A channel is a data queue that forwards the source data to the sink destination. It can be thought of as a buffer that manages input (source) and output (sink) flow rates.

■ *Sink.* The sink delivers data to destination such as HDFS, a local file, or another Flume agent.

A Flume agent must have all three of these components defined. A Flume agent can have several sources, channels, and sinks. Sources can write to multiple channels, but a sink can take data from only a single channel. Data written to a channel remain in the channel until a sink removes the data. By default, the data in a channel are kept in memory but may be optionally stored on disk to prevent data loss in the event of a network failure.

As shown in Figure 7.4, Sqoop agents may be placed in a pipeline, possibly to traverse several machines or domains. This configuration is normally used when data are collected on one machine (e.g., a web server) and sent to another machine that has access to HDFS.

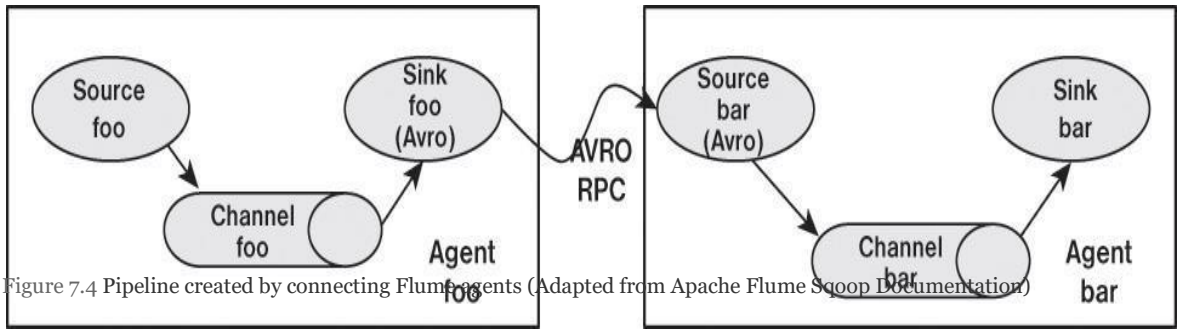


Figure 7.4 Pipeline created by connecting Flume agents (Adapted from Apache Flume Sqoop Documentation)

In a Flume pipeline, the sink from one agent is connected to the source of another. The data transfer format normally used by Flume, which is called Apache Avro, provides several useful features. First, Avro is a data serialization/deserialization system that uses a compact

binary format. The schema is sent as part of the data exchange and is defined using JSON (JavaScript Object Notation). Avro also uses remote procedure calls (RPCs) to send data. That is, an Avro sink will contact an Avro source to send data.

Another useful Flume configuration is shown in Figure 7.5. In this configuration, Flume is used to consolidate several data sources before committing them to HDFS.

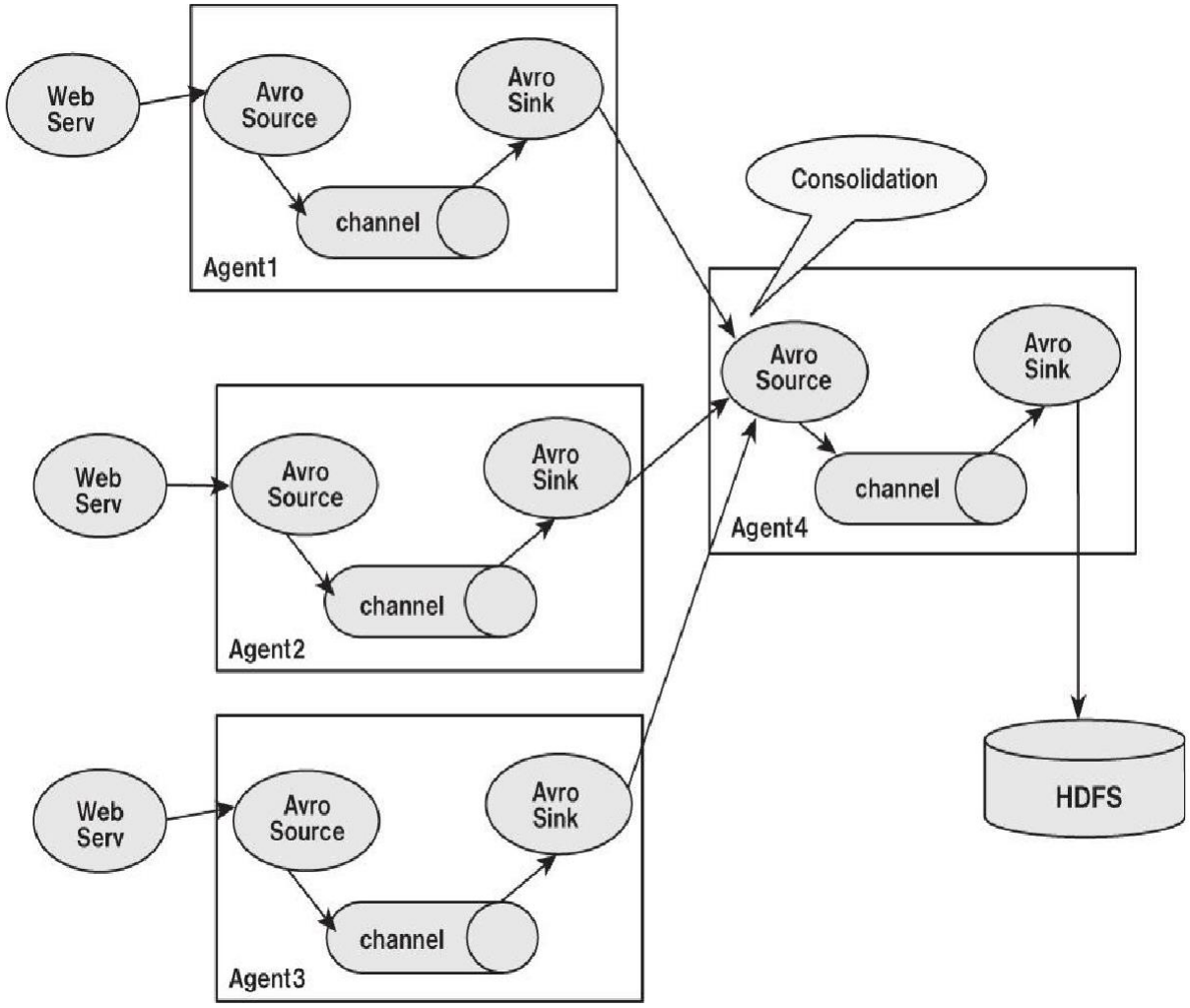


Figure 7.5 A Flume consolidation network (Adapted from Apache Flume Documentation)

There are many possible ways to construct Flume transport networks. In addition, other Flume features not described in depth here include plug-ins and interceptors that can enhance Flume pipelines.

Flume Example Walk-Through

Follow these steps to walk through a Flume example.

Step 1: Download and Install Apache Flume

Step 2: Simple Test

A simple test of Flume can be done on a single machine. To start the Flume agent, enter the flume-ng command shown here. This command uses the simple-example.conf file to configure the agent.

```
$ flume-ng agent --conf conf --conf-file simple-example.conf --name simple_agent -
Dflume.root.logger=INFO,console
```

In another terminal window, use telnet to contact the agent:

```
$ telnet localhost 44444
 Trying ::1...
 telnet: connect to address ::1: Connection refused
 Trying 127.0.0.1...
 Connected to localhost.
 Escape character is '^]'.
 testing 1 2 3
 OK
```

If Flume is working correctly, the window where the Flume agent was started will show the testing message entered in the telnet window:

Step 3: Weblog Example

In this example, a record from the weblogs from the local machine (Ambari output) will be placed into HDFS using Flume. This example is easily modified to use other weblogs from different machines. Two files are needed to configure Flume. (See the sidebar and Appendix A for file downloading instructions.)

- `web-server-target-agent.conf`—the target Flume agent that writes the data to HDFS
- `web-server-source-agent.conf`—the source Flume agent that captures the weblog data

The weblog is also mirrored on the local file system by the agent that writes to HDFS. To run the example, create the directory as root:

```
# mkdir /var/log/flume-hdfs
# chown hdfs:hadoop /var/log/flume-hdfs/
```

Next, as user hdfs, make a Flume data directory in HDFS:

```
$ hdfs dfs -mkdir /user/hdfs/flume-channel/
```

Now that you have created the data directories, you can start the Flume target agent (execute as user hdfs):

```
$ flume-ng agent -c conf -f web-server-target-agent.conf -n collector
```

This agent writes the data into HDFS and should be started before the source agent. (The source reads the weblogs.) This configuration enables automatic use of the Flume agent. The /etc/flume/conf/{flume.conf, flume-env.sh.template} files need to be configured for this purpose. For this example, the /etc/flume/conf/flume.conf file can be the same as the web-server-target.conf file (modified for your environment).

In this example, the source agent is started as root, which will start to feed the weblog data to the target agent. Alternatively, the source agent can be on another machine if desired.

flume-ng agent -c conf -f web-server-source-agent.conf -n source_agent

To see if Flume is working correctly, check the local log by using the tail command. Also confirm that the flume-ng agents are not reporting any errors (the file name will vary).

$ **tail -f /var/log/flume-hdfs/1430164482581-1**

The contents of the local log under flume-hdfs should be identical to that written into HDFS. You can inspect this file by using the hdfs -tail command (the file name will vary). Note that while running Flume, the most recent file in HDFS may have the extension .tmp appended to it. The .tmpindicates that the file is still being written by Flume. The target agent can be configured to write the file (and start another .tmp file) by setting some or all of the rollCount, rollSize, rollInterval, idleTimeout, and batchSize options in the configuration file.

$ hdfs dfs -tail flume-channel/apache_access_combined/150427/FlumeData.1430164801381

Both files should contain the same data. For instance, the preceding example had the following data in both files:

10.0.0.1 - - [27/Apr/2015:16:04:21 -0400] "GET /ambarinagios/nagios/
nagios_alerts.php?q1=alerts&alert_type=all HTTP/1.1" 200 30801 "-" "Java/1.7.0_65"
10.0.0.1 - - [27/Apr/2015:16:04:25 -0400] "POST /cgi-bin/rrd.py HTTP/1.1" 200 784
"-" "Java/1.7.0_65"
10.0.0.1 - - [27/Apr/2015:16:04:25 -0400] "POST /cgi-bin/rrd.py HTTP/1.1" 200 508
"-" "Java/1.7.0_65"

MANAGE HADOOP WORKFLOWS WITH APACHE OOZIE

Oozie is a workflow director system designed to run and manage multiple related Apache Hadoop jobs. For instance, complete data input and analysis may require several discrete Hadoop jobs to be run as a workflow in which the output of one job serves as the input for a successive job. Oozie is designed to construct and manage these workflows. Oozie is not a substitute for the YARN scheduler. That is, YARN manages resources for individual Hadoop jobs, and Oozie provides a way to connect and control Hadoop jobs on the cluster.

Oozie workflow jobs are represented as directed acyclic graphs (DAGs) of actions. (DAGs are basically graphs that cannot have directed loops.) Three types of Oozie jobs are permitted:

■ Workflow—a specified sequence of Hadoop jobs with outcome-based decision points and control dependency. Progress from one action to another cannot happen until the first action is complete.

■ Coordinator—a scheduled workflow job that can run at various time intervals or when data become available.

■ Bundle—a higher-level Oozie abstraction that will batch a set of coordinator jobs.

Oozie is integrated with the rest of the Hadoop stack, supporting several types of Hadoop jobs out of the box (e.g., Java MapReduce, Streaming MapReduce, Pig, Hive, and Sqoop) as well as system-specific jobs (e.g., Java programs and shell scripts). Oozie also provides a CLI and a web UI for monitoring jobs.

Figure 7.6 depicts a simple Oozie workflow. In this case, Oozie runs a basic MapReduce operation. If the application was successful, the job ends; if an error occurred, the job is killed.

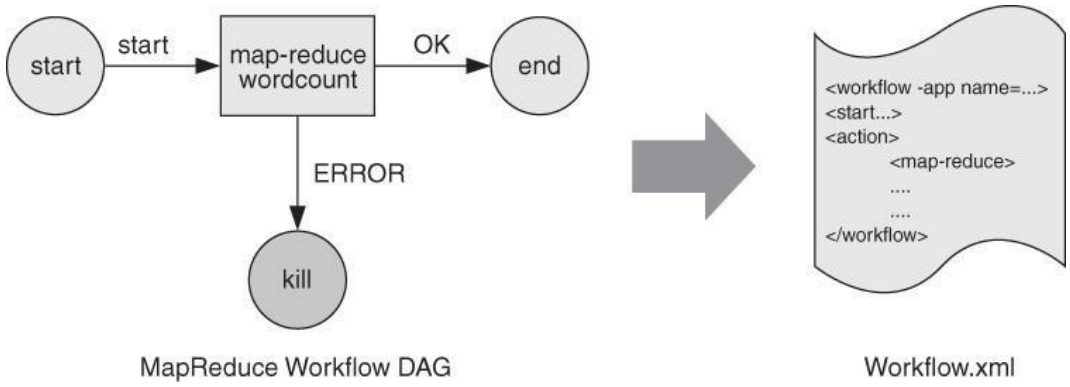


Figure 7.6 A simple Oozie DAG workflow (Adapted from Apache Oozie Documentation)

Oozie workflow definitions are written in hPDL (an XML Process Definition Language). Such workflows contain several types of nodes:

■ **Control flow nodes** define the beginning and the end of a workflow. They include start, end, and optional fail nodes.

■ **Action nodes** are where the actual processing tasks are defined. When an action node finishes, the remote systems notify Oozie and the next node in the workflow is executed. Action nodes can also include HDFS commands.

■ **Fork/join nodes** enable parallel execution of tasks in the workflow. The fork node enables two or more tasks to run at the same time. A join node represents a rendezvous point that must wait until all forked tasks complete.

■ **Control flow nodes** enable decisions to be made about the previous task. Control decisions are based on the results of the previous action (e.g., file size or file existence). Decision nodes are essentially switch-case statements that use JSP EL (Java Server Pages—Expression Language) that evaluate to either true or false.

Figure 7.7 depicts a more complex workflow that uses all of these node types.

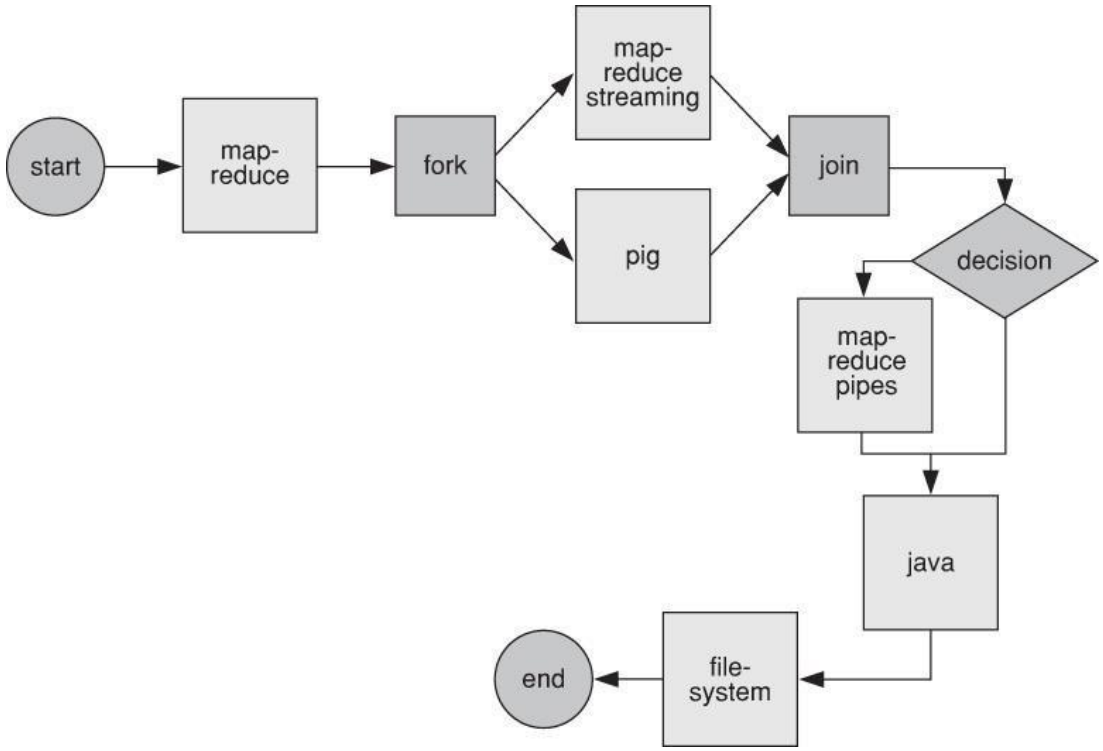


Figure 7.7 A more complex Oozie DAG workflow (Adapted from Apache Oozie Documentation)

Oozie Example Walk-Through

Step 1: Download Oozie Examples

The Oozie examples used in this section can be found on the book website (see Appendix A). They are also available as part of the oozie-client.noarch RPM in the Hortonworks HDP 2.x packages. For HDP 2.1, the following command can be used to extract the files into the working directory used for the demo:

$ **tar xvzf /usr/share/doc/oozie-4.0.0.2.1.2.1/oozie-examples.tar.gz**

For HDP 2.2, the following command will extract the files:

$ **tar xvzf /usr/hdp/2.2.4.2-2/oozie/doc/oozie-examples.tar.gz**

Once extracted, rename the examples directory to oozie-examples so that you will not confuse it with the other examples directories.

$ **mv examples oozie-examples**

The examples must also be placed in HDFS. Enter the following command to move the example files into HDFS:

$ **hdfs dfs -put oozie-examples/ oozie-examples**

The Oozie shared library must be installed in HDFS. If you are using the Ambari installation of HDP 2.x, this library is already found in HDFS: /user/oozie/share/lib.

Step 2: Run the Simple MapReduce Example

Move to the simple MapReduce example directory:

$ **cd oozie-examples/apps/map-reduce/**

This directory contains two files and a lib directory. The files are:

- The job.properties file defines parameters (e.g., path names, ports) for a job. This file may change per job.
- The workflow.xml file provides the actual workflow for the job. In this case, it is a simple MapReduce (pass/fail). This file usually stays the same between jobs.

The job.properties file included in the examples requires a few edits to work properly. Using a text editor, change the following lines by adding the host name of the NameNode and ResourceManager (indicated by jobTracker in the file).

As shown in Figure 7.6, this simple workflow runs an example MapReduce job and prints an error message if it fails.

To run the Oozie MapReduce example job from the oozie-examples/apps/map- reduce directory, enter the following line:

$ **oozie job -run -oozie http://limulus:11000/oozie -config job.properties**

When Oozie accepts the job, a job ID will be printed:

job: 0000001-150424174853048-oozie-oozi-W

You will need to change the "limulus" host name to match the name of the node running your Oozie server. The job ID can be used to track and control job progress.

To avoid having to provide the -oozie option with the Oozie URL every time you run the ooziecommand, set the OOZIE_URL environment variable as follows (using your Oozie server host name in place of "limulus"):

$ **export OOZIE_URL="http://limulus:11000/oozie"**

You can now run all subsequent Oozie commands without specifying the -oozie URL option. For instance, using the job ID, you can learn about a particular job's progress by issuing the following command:

$ **oozie job -info 0000001-150424174853048-oozie-oozi-W**

The resulting output (line length compressed) is shown in the following listing. Because this job is just a simple test, it may be complete by the time you issue the -info command. If it is not complete, its progress will be indicated in the listing.

```
Job ID : 0000001-150424174853048-oozie-oozi-W
------------------------------------------------------------------------------------
Workflow Name : map-reduce-wf
App Path      : hdfs://limulus:8020/user/hdfs/examples/apps/map-reduce
Status        : SUCCEEDED
Run           : 0
User          : hdfs
Group         : -
Created       : 2015-04-29 20:52 GMT
Started       : 2015-04-29 20:52 GMT
Last Modified : 2015-04-29 20:53 GMT
Ended         : 2015-04-29 20:53 GMT
CoordAction ID: -

Actions
------------------------------------------------------------------------------------
ID                                Status   Ext ID    Ext Status Err Code
------------------------------------------------------------------------------------
0000001-150424174853048-oozie
  -oozi-W@:start:                 OK       -         OK         -
------------------------------------------------------------------------------------
0000001-150424174853048-oozie
  -oozi-W@mr-node                 OK job_1429912013449_0006 SUCCEEDED -
------------------------------------------------------------------------------------
0000001-150424174853048-oozie
  -oozi-W@end                     OK       -         OK         -
------------------------------------------------------------------------------------
```

The various steps shown in the output can be related directly to the workflow.xml mentioned previously. Note that the MapReduce job number is provided. This job will also be listed in the ResourceManager web user interface. The application output is located in HDFS under the oozie-examples/output-data/map-reduce directory.

Step 3: Run the Oozie Demo Application

A more sophisticated example can be found in the demo directory (oozie-examples/apps/demo). This workflow includes MapReduce, Pig, and file system tasks as well as fork, join, decision, action, start, stop, kill, and end nodes.

Move to the demo directory and edit the job.properties file as described previously. Entering the following command runs the workflow (assuming the OOZIE_URL environment variable has been set):

$ **oozie job -run -config job.properties**

You can track the job using either the Oozie command-line interface or the Oozie web console. To start the web console from within Ambari, click on the Oozie service, and then click on the Quick Links pull-down menu and select Oozie Web UI. Alternatively, you can start the Oozie web UI by connecting to the Oozie server directly. For example, the following command will bring up the Oozie UI (use your Oozie server host name in place of "limulus"):

$ **firefox http://limulus:11000/oozie/**

Figure 7.8 shows the main Oozie console window.

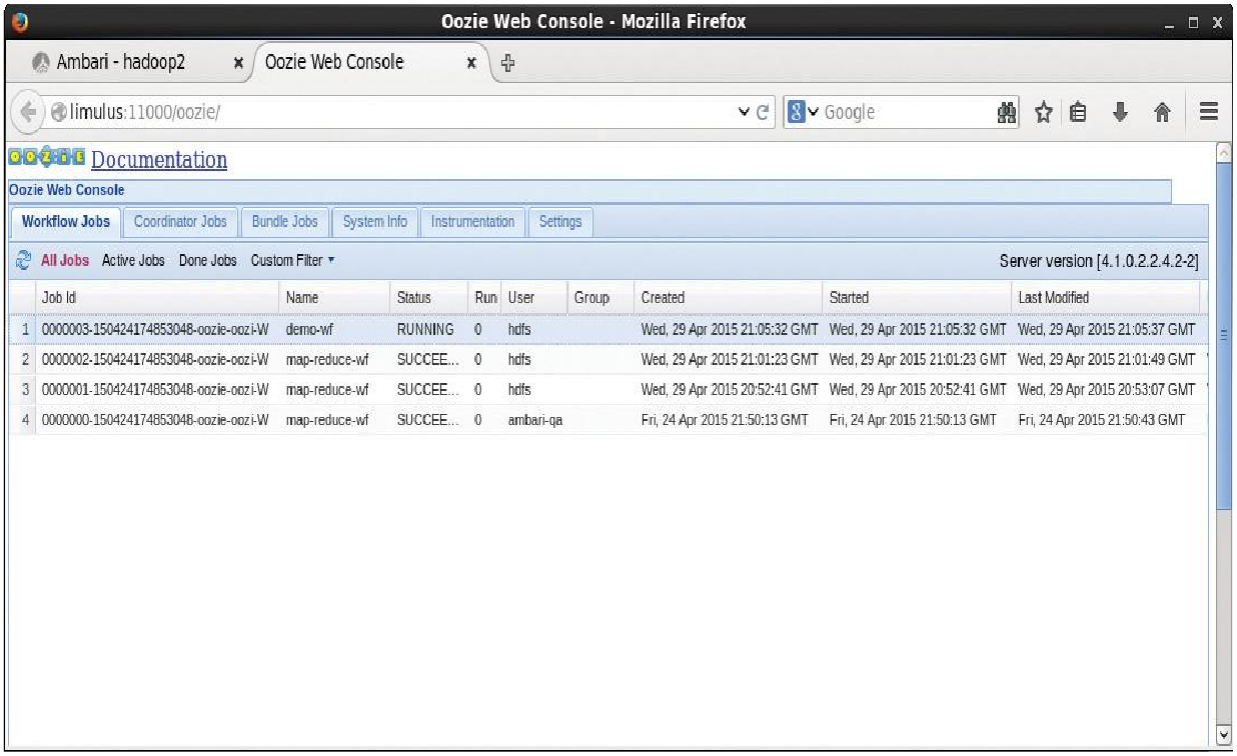


Figure 7.8 Oozie main console window

Workflow jobs are listed in tabular form, with the most recent job appearing first. If you click on a workflow, the Job Info window in Figure 7.9 will be displayed. The job progression results, similar to those printed by the Oozie command line, are shown in the Actions window at the bottom.

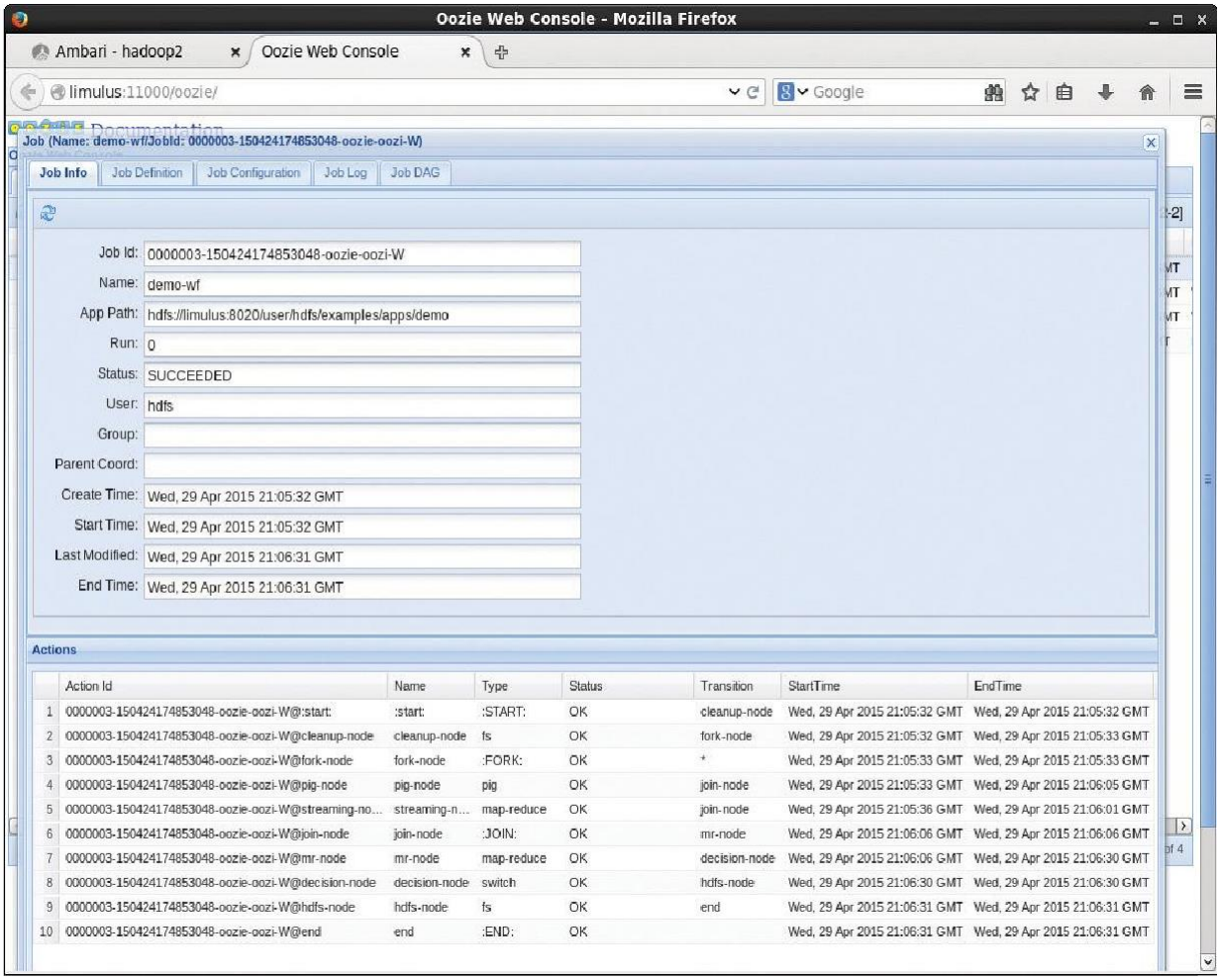


Figure 7.9 Oozie workflow information window

Other aspects of the job can be examined by clicking the other tabs in the window. The last tab actually provides a graphical representation of the workflow DAG. If the job is not complete, it will highlight the steps that have been completed thus far. The DAG for the demo example is shown in Figure 7.10. The actual image was split to fit better on the page. As with the previous example, comparing this information to workflow.xml file can provide further insights into how Oozie operates.

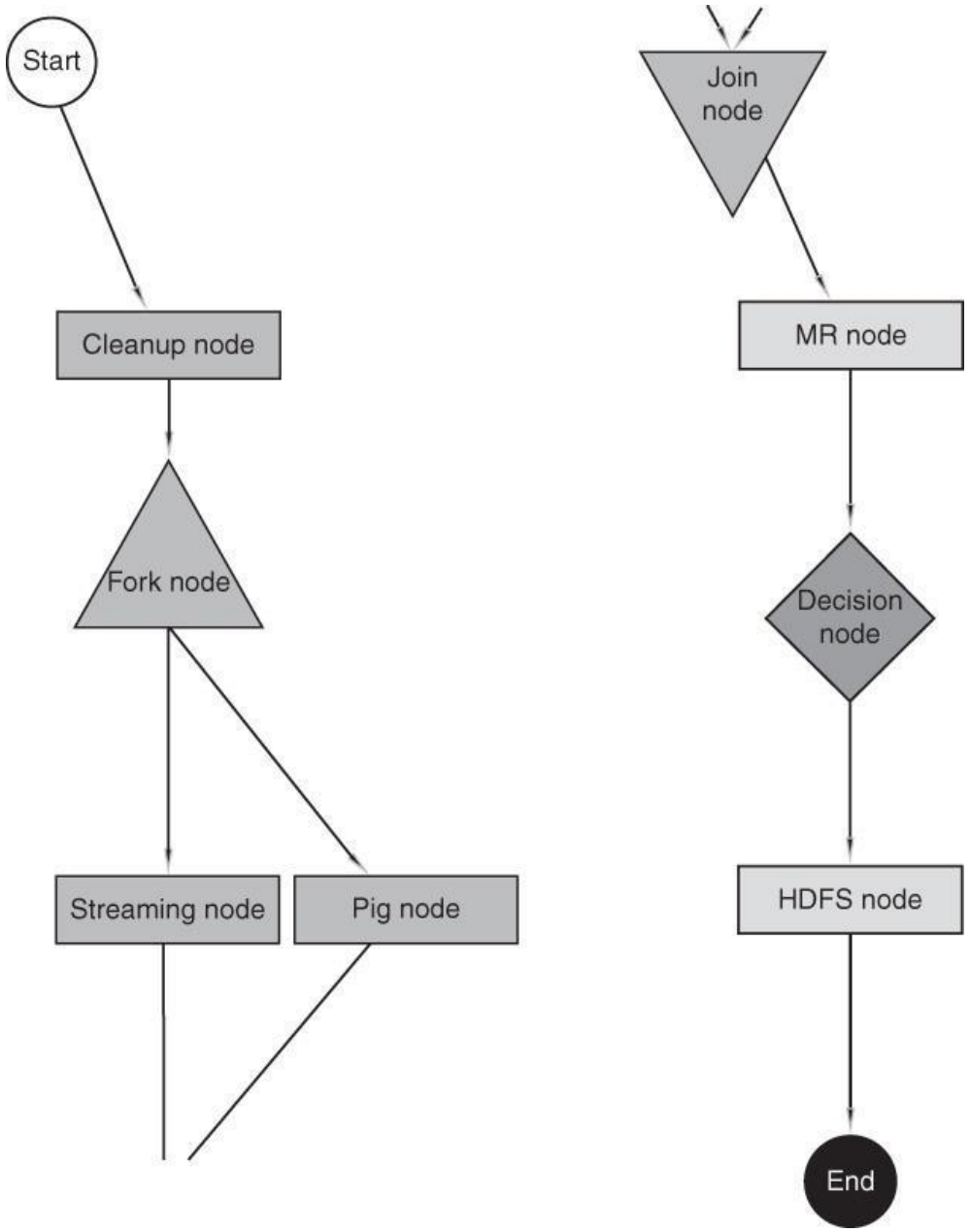


Figure 7.10 Oozie-generated workflow DAG for the demo example, as it appears on the screen

A Short Summary of Oozie Job Commands

The following summary lists some of the more commonly encountered Oozie commands. See the latest documentation at http://oozie.apache.org for more information. (Note that the examples here assume OOZIE_URL is defined.)

■ Run a workflow job (returns `_OOZIE_JOB_ID_`):

```
$ oozie job -run -config JOB_PROPERITES
```

■ Submit a workflow job (returns `_OOZIE_JOB_ID_` but does not start):

```
$ oozie job -submit -config JOB_PROPERTIES
```

- Start a submitted job:

```
$ oozie job -start _OOZIE_JOB_ID_
```

- Check a job's status:

```
$ oozie job -info _OOZIE_JOB_ID_
```

- Suspend a workflow:

```
$ oozie job -suspend _OOZIE_JOB_ID_
```

- Resume a workflow:

```
$ oozie job -resume _OOZIE_JOB_ID_
```

- Rerun a workflow:

```
$ oozie job -rerun _OOZIE_JOB_ID_ -config JOB_PROPERTIES
```

- Kill a job:

```
$ oozie job -kill _OOZIE_JOB_ID_
```

- View server logs:

```
$ oozie job  -logs _OOZIE_JOB_ID_
```

Full logs are available at /var/log/oozie on the Oozie server.

USING APACHE HBASE

Apache HBase is an open source, distributed, versioned, nonrelational database modeled after Google's Bigtable. Like Bigtable, HBase leverages the distributed data storage provided by the underlying distributed file systems spread across commodity servers. Apache HBase provides Bigtable-like capabilities on top of Hadoop and HDFS. Some of the more important features include the following capabilities:

- Linear and modular scalability
- Strictly consistent reads and writes
- Automatic and configurable sharding of tables
- Automatic failover support between RegionServers
- Convenient base classes for backing Hadoop MapReduce jobs with Apache HBase tables
- Easy-to-use Java API for client access

HBase Data Model Overview

A table in HBase is similar to other databases, having rows and columns. Columns in HBase are grouped into column families, all with the same prefix. For example, consider a table of daily stock prices. There may be a column family called “price” that has four members—price:open, price:close, price:low, and price:high. A column does not need to be a family. For instance, the stock table may have a column named “volume” indicating how many shares were traded. All column family members are stored together in the physical file system.

Specific HBase cell values are identified by a row key, column (column family and column), and version (timestamp). It is possible to have many versions of data within an HBase cell. A version is specified as a timestamp and is created each time data are written to a cell. Almost anything can serve as a row key, from strings to binary representations of longs to serialized data structures. Rows are lexicographically sorted with the lowest order appearing first in a table. The empty byte array denotes both the start and the end of a table’s namespace. All table accesses are via the table row key, which is considered its primary key.

HBase Example Walk-Through

HBase provides a shell for interactive use. To enter the shell, type the following as a user:

```
$ hbase shell
hbase(main):001:0>
```

To exit the shell, type **exit**.

Various commands can be conveniently entered from the shell prompt. For instance, the status command provides the system status:

```
hbase(main):001:0> status
4 servers, 0 dead, 1.0000 average load
```

Additional arguments can be added to the status command, including 'simple', 'summary', or 'detailed'. The single quotes are needed for proper operation. For example, the following command will provide simple status information for the four HBase servers (actual server statistics have been removed for clarity):

```
hbase(main):002:0> status 'simple'
4 live servers
   n1:60020 1429912048329
     ...
   n2:60020 1429912040653
     ...
   limulus:60020 1429912041396
     ...
   n0:60020 1429912042885
     ...
0 dead servers
Aggregate load: 0, regions: 4
```

Other basic commands, such as `version` or `whoami`, can be entered directly at the hbase(main)prompt. In the example that follows, we will use a small set of daily stock price data for Apple computer. The data have the following form:

Date	Open	High	Low	Close	Volume
6-May-15	126.56	126.75	123.36	125.01	71820387

The data can be downloaded from Google using the following command. Note that other stock prices are available by changing the NASDAQ:AAPL argument to any other valid exchange and stock name (e.g., NYSE: IBM).

$ **wget -O Apple-stock.csv**
http://www.google.com/finance/historical?q=NASDAQ:AAPL\&authuser=0\&output=csv

The Apple stock price database is in comma-separated format (csv) and will be used to illustrate some basic operations in the HBase shell.

Create the Database

The next step is to create the database in HBase using the following command:

```
hbase(main):006:0> create 'apple', 'price' , 'volume'
0 row(s) in 0.8150 seconds
```

In this case, the table name is apple, and two columns are defined. The date will be used as the row key. The price column is a family of four values (open, close, low, high). The put command is used to add data to the database from within the shell. For instance, the preceding data can be entered by using the following commands:

```
put 'apple','6-May-15','price:open','126.56'
put 'apple','6-May-15','price:high','126.75'
put 'apple','6-May-15','price:low','123.36'
put 'apple','6-May-15','price:close','125.01'
put 'apple','6-May-15','volume','71820387'
```

The shell also keeps a history for the session, and previous commands can be retrieved and edited for resubmission.

Inspect the Database

The entire database can be listed using the scan command. Be careful when using this command with large databases. This example is for one row.

```
hbase(main):006:0> scan 'apple'
ROW          COLUMN+CELL
 6-May-15      column=price:close, timestamp=1430955128359, value=125.01
 6-May-15      column=price:high, timestamp=1430955126024, value=126.75
 6-May-15      column=price:low, timestamp=1430955126053, value=123.36
 6-May-15      column=price:open, timestamp=1430955125977, value=126.56
 6-May-15      column=volume:, timestamp=1430955141440, value=71820387
```

Get a Row

You can use the row key to access an individual row. In the stock price database, the date is the row key.

```
hbase(main):008:0> get 'apple', '6-May-15'
COLUMN                  CELL
 price:close            timestamp=1430955128359, value=125.01
 price:high             timestamp=1430955126024, value=126.75
 price:low              timestamp=1430955126053, value=123.36
 price:open             timestamp=1430955125977, value=126.56
 volume:                timestamp=1430955141440, value=71820387
5 row(s) in 0.0130 seconds
```

Get Table Cells

A single cell can be accessed using the get command and the COLUMN option:

```
hbase(main):013:0> get 'apple', '5-May-15', {COLUMN => 'price:low'}
COLUMN                  CELL
 price:low              timestamp=1431020767444, value=125.78
1 row(s) in 0.0080 seconds
```

In a similar fashion, multiple columns can be accessed as follows:

```
hbase(main):012:0> get 'apple', '5-May-15', {COLUMN => ['price:low', 'price:high']}
COLUMN                  CELL
 price:high             timestamp=1431020767444, value=128.45
 price:low              timestamp=1431020767444, value=125.78
2 row(s) in 0.0070 seconds
```

Delete a Cell

A specific cell can be deleted using the following command:

```
hbase(main):009:0> delete 'apple', '6-May-15' , 'price:low'
```

If the row is inspected using get, the price:low cell is not listed.

```
hbase(main):010:0> get 'apple', '6-May-15'
COLUMN                  CELL
 price:close            timestamp=1430955128359, value=125.01
 price:high             timestamp=1430955126024, value=126.75
 price:open             timestamp=1430955125977, value=126.46
 volume:                timestamp=1430955141440, value=71820387
4 row(s) in 0.0130 seconds
```

Delete a Row

You can delete an entire row by giving the deleteall command as follows:

```
hbase(main):009:0> deleteall 'apple', '6-May-15'
```

Remove a Table

To remove (drop) a table, you must first disable it. The following two commands remove the appletable from Hbase:

```
hbase(main):009:0> disable 'apple'
hbase(main):010:0> drop 'apple'
```

Scripting Input

Commands to the HBase shell can be placed in bash scripts for automated processing. For instance, the following can be placed in a bash script:

```
echo "put 'apple','6-May-15','price:open','126.56'" | hbase shell
```

The book software page includes a script (input_to_hbase.sh) that imports the Apple- stock.csv file into HBase using this method. It also removes the column titles in the first line. The script will load the entire file into HBase when you issue the following command:

```
$ input_to_hbase.sh Apple-stock.csv
```

While the script can be easily modified to accommodate other types of data, it is not recommended for production use because the upload is very inefficient and slow. Instead, this script is best used to experiment with small data files and different types of data.

Adding Data in Bulk

There are several ways to efficiently load bulk data into HBase. Covering all of these methods is beyond the scope of this chapter. Instead, we will focus on the ImportTsv utility, which loads data in tab-separated values (tsv) format into HBase. It has two distinct usage modes:

- Loading data from a tsv-format file in HDFS into HBase via the put command
- Preparing StoreFiles to be loaded via the completebulkload utility

The following example shows how to use ImportTsv for the first option, loading the tsv- format file using the put command.

The first step is to convert the Apple-stock.csv file to tsv format. The following script, which is included in the book software, will remove the first line and do the conversion. In doing so, it creates a file named Apple-stock.tsv.

```
$ convert-to-tsv.sh Apple-stock.csv
```

Next, the new file is copied to HDFS as follows:

```
$ hdfs dfs -put Apple-stock.tsv /tmp
```

Finally, ImportTsv is run using the following command line. Note the column designation in the -Dimporttsv.columns option. In the example, the HBASE_ROW_KEY is set as the first column—that is, the date for the data.

```
$ hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -
Dimporttsv.columns=HBASE_ROW_KEY,price:open,price:high,price:low,price:close,volume apple
/tmp/Apple-stock.tsv
```

The ImportTsv command will use MapReduce to load the data into HBase. To verify that the command works, drop and re-create the apple database, as described previously, before running the import command.

8. Hadoop YARN Applications

In This Chapter:

- The YARN Distributed-Shell is introduced as a non-MapReduce application.
- The Hadoop YARN application and operation structure is explained.
- A summary of YARN application frameworks is provided.

YARN DISTRIBUTED-SHELL

The Hadoop YARN project includes the Distributed-Shell application, which is an example of a Hadoop non-MapReduce application built on top of YARN. Distributed-Shell is a simple mechanism for running shell commands and scripts in containers on multiple nodes in a Hadoop cluster. This application is not meant to be a production administration tool, but rather a demonstration of the non-MapReduce capability that can be implemented on top of YARN. There are multiple mature implementations of a distributed shell that administrators typically use to manage a cluster of machines.

In addition, Distributed-Shell can be used as a starting point for exploring and building Hadoop YARN applications. This chapter offers guidance on how the Distributed-Shell can be used to understand the operation of YARN applications.

USING THE YARN DISTRIBUTED-SHELL

For the purpose of the examples presented in the remainder of this chapter, we assume and assign the following installation path, based on Hortonworks HDP 2.2, the Distributed-Shell application:

```
$ export YARN_DS=/usr/hdp/current/hadoop-yarn-client/hadoop-yarn-applications-
distributedshell.jar
```

For the pseudo-distributed install using Apache Hadoop version 2.6.0, the following path will run the Distributed-Shell application (assuming $HADOOP_HOME is defined to reflect the location Hadoop):

$ **export YARN_DS=$HADOOP_HOME/share/hadoop/yarn/hadoop-yarn-applications-distributedshell-2.6.0.jar**

If another distribution is used, search for the file hadoop-yarn-applications-distributedshell*.jar and set $YARN_DS based on its location. Distributed-Shell exposes various options that can be found by running the following command:

$ **yarn org.apache.hadoop.yarn.applications.distributedshell.Client -jar $YARN_DS -help**

The output of this command follows; we will explore some of these options in the examples illustrated in this chapter.

```
usage: Client
 -appname <arg>                Application Name. Default value – DistributedShell
 -container_memory <arg>       Amount of memory in MB to be requested to run the shell command
 -container_vcores <arg>       Amount of virtual cores to be requested to run the shell command
 -create                       Flag to indicate whether to create the domain specified with -domain.
 -debug                        Dump out debug information
 -domain <arg>                 ID of the timeline domain where the timeline entities will be put
 -help                         Print usage
 -jar <arg>                    Jar file containing the application master
 -log_properties <arg>         log4j.properties file
 -master_memory <arg>          Amount of memory in MB to be requested to run the application master
 -master_vcores <arg>          Amount of virtual cores to be requested to run the application master
 -modify_acls <arg>            Users and groups that allowed to modify the timeline entities in the given domain
 -timeout <arg>                Application timeout in milliseconds
 -view_acls <arg>              Users and groups that allowed to view the timeline entities in the given domain
```

A Simple Example

The simplest use-case for the Distributed-Shell application is to run an arbitrary shell command in a container. We will demonstrate the use of the uptime command as an example. This command is run on the cluster using Distributed-Shell as follows:

$ **yarn org.apache.hadoop.yarn.applications.distributedshell.Client -jar $YARN_DS -shell_command uptime**

By default, Distributed-Shell spawns only one instance of a given shell command. When this command is run, you can see progress messages on the screen but nothing about the actual shell command. If the shell command succeeds, the following should appear at the end of the output:

```
15/05/27 14:48:53 INFO distributedshell.Client: Application completed successfully
```

If the shell command did not work for whatever reason, the following message will be displayed:

```
15/05/27 14:58:42 ERROR distributedshell.Client: Application failed to complete
successfully
```

The next step is to examine the output for the application. Distributed-Shell redirects the output of the individual shell commands run on the cluster nodes into the log files, which are found either on the individual nodes or aggregated onto HDFS, depending on whether log aggregation is enabled.

Assuming log aggregation is enabled, the results for each instance of the command can be found by using the yarn logs command. For the previous uptime example, the following command can be used to inspect the logs:

```
$ yarn logs -applicationId application_1432831236474_0001
```

The abbreviated output follows:

```
Container: container_1432831236474_0001_01_000001 on n0_45454
==========================================================
LogType:AppMaster.stderr
Log Upload Time:Thu May 28 12:41:58 -0400 2015
LogLength:3595
Log Contents:
15/05/28 12:41:52 INFO distributedshell.ApplicationMaster: Initializing
ApplicationMaster
[...]
Container: container_1432831236474_0001_01_000002 on n1_45454
==========================================================
LogType:stderr
Log Upload Time:Thu May 28 12:41:59 -0400 2015
LogLength:0
Log Contents:

LogType:stdout
Log Upload Time:Thu May 28 12:41:59 -0400 2015
LogLength:71
Log Contents:
 12:41:56 up 33 days, 19:28, 0 users, load average: 0.08, 0.06, 0.01
```

Notice that there are two containers. The first container (`con..._000001`) is the ApplicationMaster for the job. The second container (con..._000002) is the actual shell script. The output for the uptime command is located in the second containers stdout after the Log Contents: label.

Using More Containers

Distributed-Shell can run commands to be executed on any number of containers by way of the -num_containers argument. For example, to see on which nodes the Distributed-Shell command was run, the following command can be used:

$ yarn org.apache.hadoop.yarn.applications.distributedshell.Client -jar $YARN_DS -shell_command hostname -num_containers 4

If we now examine the results for this job, there will be five containers in the log. The four command containers (2 through 5) will print the name of the node on which the container was run.

Distributed-Shell Examples with Shell Arguments

Arguments can be added to the shell command using the -shell_args option. For example, to do a ls -l in the directory from where the shell command was run, we can use the following commands:

$ yarn org.apache.hadoop.yarn.applications.distributedshell.Client -jar $YARN_DS -shell_command ls -shell_args -l

The resulting output from the log file is as follows:

```
total 20
-rw-r--r-- 1 yarn hadoop 74 May 28 10:37 container_tokens
-rwx------ 1 yarn hadoop 643 May 28 10:37 default_container_executor_session.sh
-rwx------ 1 yarn hadoop 697 May 28 10:37 default_container_executor.sh
-rwx ------1 yarn hadoop 1700 May 28 10:37 launch_container.sh
drwx--x ---2 yarn hadoop 4096 May 28 10:37 tmp
```

As can be seen, the resulting files are new and not located anywhere in HDFS or the local file system. When we explore further by giving a pwd command for Distributed-Shell, the following directory is listed and created on the node that ran the shell command:

/hdfs2/hadoop/yarn/local/usercache/hdfs/appcache/application_1432831236474_0003/container_1432831236474_0003_01_000002/

Searching for this directory will prove to be problematic because these transient files are used by YARN to run the Distributed-Shell application and are removed once the application finishes. You can preserve these files for a specific interval by adding the following lines to the yarn-site.xmlconfiguration file and restarting YARN:

```
<property>
   <name>yarn.nodemanager.delete.debug-delay-sec</name>
   <value>100000</value>
</property>
```

Choose a delay, in seconds, to preserve these files, and remember that all applications will create these files. If you are using Ambari, look on the YARN Configs tab under the Advanced yarn-site options, make the change and restart YARN. (See Chapter 9, "Managing Hadoop with Apache Ambari," for more information on Ambari administration.) These files will be retained on the individual nodes only for the duration of the specified delay.

When debugging or investigating YARN applications, these files—in particular, launch_container.sh—offer important information about YARN processes. Distributed-Shell can be used to see what this file contains. Using DistributedShell, the contents of the launch_container.sh file can be printed with the following command:

$ yarn org.apache.hadoop.yarn.applications.distributedshell.Client -jar $YARN_DS -shell_command cat -shell_args launch_container.sh

This command prints the launch_container.sh file that is created and run by YARN. The contents of this file are shown in Listing 8.1. The file basically exports some important YARN variables and then, at the end, "execs" the command (cat launch_container.sh) directly and sends any output to logs.

Listing 8.1 **Distributed-Shell launch_container.sh File**

```
#!/bin/bash

export NM_HTTP_PORT="8042"
export LOCAL_DIRS="/opt/hadoop/yarn/local/usercache/hdfs/appcache/
application_1432816241597_0004,/hdfs1/hadoop/yarn/local/usercache/hdfs/appc
ache/
application_1432816241597_0004,/hdfs2/hadoop/yarn/local/usercache/hdfs/appc
ache/
application_1432816241597_0004"
export JAVA_HOME="/usr/lib/jvm/java-1.7.0-openjdk.x86_64"
export
NM_AUX_SERVICE_mapreduce_shuffle="AAA0+gAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAA=
"
export HADOOP_YARN_HOME="/usr/hdp/current/hadoop-yarn-client"
export HADOOP_TOKEN_FILE_LOCATION="/hdfs2/hadoop/yarn/local/usercache/hdfs/
appcache/application_1432816241597_0004/container_1432816241597_0004_01_000
002/
container_tokens"
export NM_HOST="limulus"
export JVM_PID="$$"
export USER="hdfs"
export PWD="/hdfs2/hadoop/yarn/local/usercache/hdfs/appcache/
application_1432816241597_0004/container_1432816241597_0004_01_000002"
export CONTAINER_ID="container_1432816241597_0004_01_000002"
export NM_PORT="45454"
export HOME="/home/"
export LOGNAME="hdfs"
export HADOOP_CONF_DIR="/etc/hadoop/conf"
```

```
export MALLOC_ARENA_MAX="4"
export LOG_DIRS="/opt/hadoop/yarn/log/application_1432816241597_0004/
container_1432816241597_0004_01_000002,/hdfs1/hadoop/yarn/log/
application_1432816241597_0004/container_1432816241597_0004_01_000002,/hdfs
2/
hadoop/yarn/log/application_1432816241597_0004/
container_1432816241597_0004_01_000002"
exec /bin/bash -c "cat launch_container.sh
1>/hdfs2/hadoop/yarn/log/application_1432816241597_0004/
container_1432816241597_0004_01_000002/stdout 2>/hdfs2/hadoop/yarn/log/
application_1432816241597_0004/container_1432816241597_0004_01_000002/stder
r "
hadoop_shell_errorcode=$?
if [ $hadoop_shell_errorcode -ne 0 ]
then
  exit $hadoop_shell_errorcode
fi
```

There are more options for the Distributed-Shell that you can test. The real value of the Distributed-Shell application is its ability to demonstrate how applications are launched within the Hadoop YARN infrastructure. It is also a good starting point when you are creating YARN applications.

STRUCTURE OF YARN APPLICATIONS

The structure and operation of a YARN application are covered briefly in this section.

The central YARN ResourceManager runs as a scheduling daemon on a dedicated machine and acts as the central authority for allocating resources to the various competing applications in the cluster. The ResourceManager has a central and global view of all cluster resources and, therefore, can ensure fairness, capacity, and locality are shared across all users. Depending on the application demand, scheduling priorities, and resource availability, the ResourceManager dynamically allocates resource containers to applications to run on particular nodes. A container is a logical bundle of resources (e.g., memory, cores) bound to a particular cluster node. To enforce and track such assignments, the ResourceManager interacts with a special system daemon running on each node called the NodeManager. Communications between the ResourceManager and NodeManagers are heartbeat based for scalability. NodeManagers are responsible for local monitoring of resource availability, fault reporting, and container life-cycle management (e.g., starting and killing jobs). The ResourceManager depends on the NodeManagers for its “global view” of the cluster.

User applications are submitted to the ResourceManager via a public protocol and go through an admission control phase during which security credentials are validated and various operational and administrative checks are performed. Those applications that are accepted

pass to the scheduler and are allowed to run. Once the scheduler has enough resources to satisfy the request, the application is moved from an accepted state to a running state. Aside from internal bookkeeping, this process involves allocating a container for the single ApplicationMaster and spawning it on a node in the cluster. Often called container 0, the ApplicationMaster does not have any additional resources at this point, but rather must request additional resources from the ResourceManager.

The ApplicationMaster is the "master" user job that manages all application life-cycle aspects, including dynamically increasing and decreasing resource consumption (i.e., containers), managing the flow of execution (e.g., in case of MapReduce jobs, running reducers against the output of maps), handling faults and computation skew, and performing other local optimizations. The ApplicationMaster is designed to run arbitrary user code that can be written in any programming language, as all communication with the ResourceManager and NodeManager is encoded using extensible network protocols

YARN makes few assumptions about the ApplicationMaster, although in practice it expects most jobs will use a higher-level programming framework. By delegating all these functions to ApplicationMasters, YARN's architecture gains a great deal of scalability, programming model flexibility, and improved user agility. For example, upgrading and testing a new MapReduce framework can be done independently of other running MapReduce frameworks.

Typically, an ApplicationMaster will need to harness the processing power of multiple servers to complete a job. To achieve this, the ApplicationMaster issues resource requests to the ResourceManager. The form of these requests includes specification of locality preferences (e.g., to accommodate HDFS use) and properties of the containers. The ResourceManager will attempt to satisfy the resource requests coming from each application according to availability and scheduling policies. When a resource is scheduled on behalf of an ApplicationMaster, the ResourceManager generates a lease for the resource, which is acquired by a subsequent ApplicationMaster heartbeat. The ApplicationMaster then works with the NodeManagers to start the resource. A token-based security mechanism guarantees its authenticity when the ApplicationMaster presents the container lease to the NodeManager. In a typical situation, running containers will communicate with the ApplicationMaster through an application-specific protocol to report status and health information and to receive framework-specific commands. In this way, YARN provides a basic infrastructure for monitoring and life-cycle management of containers, while each framework manages

application-specific semantics independently. This design stands in sharp contrast to the original Hadoop version 1 design, in which scheduling was designed and integrated around managing only MapReduce tasks.

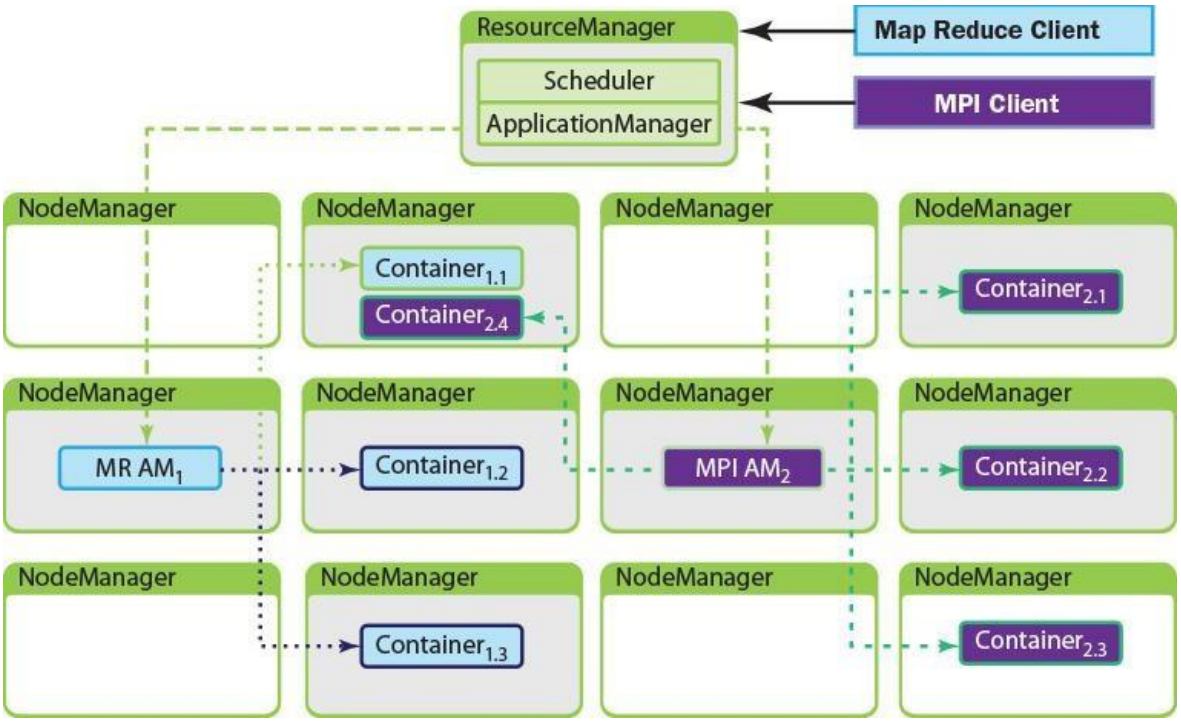


Figure 8.1 YARN architecture with two clients (MapReduce and MPI). The darker client (MPI AM_2) is running an MPI application, and the lighter client (MR AM_1) is running a MapReduce application. (From Arun C. Murthy, et al., *Apache Hadoop™ YARN*, copyright © 2014, p. 45. Reprinted and electronically reproduced by permission of Pearson Education, Inc., New York, NY.)

YARN APPLICATION FRAMEWORKS

One of the most exciting aspects of Hadoop version 2 is the capability to run all types of applications on a Hadoop cluster. In Hadoop version 1, the only processing model available to users is MapReduce. In Hadoop version 2, MapReduce is separated from the resource management layer of Hadoop and placed into its own application framework. Indeed, the growing number of YARN applications offers a high level and multifaceted interface to the Hadoop data lake.

YARN presents a resource management platform, which provides services such as scheduling, fault monitoring, data locality, and more to MapReduce and other frameworks. Figure 8.2 illustrates some of the various frameworks that will run under YARN. Note that the Hadoop version 1 applications (e.g., Pig and Hive) run under the MapReduce framework.

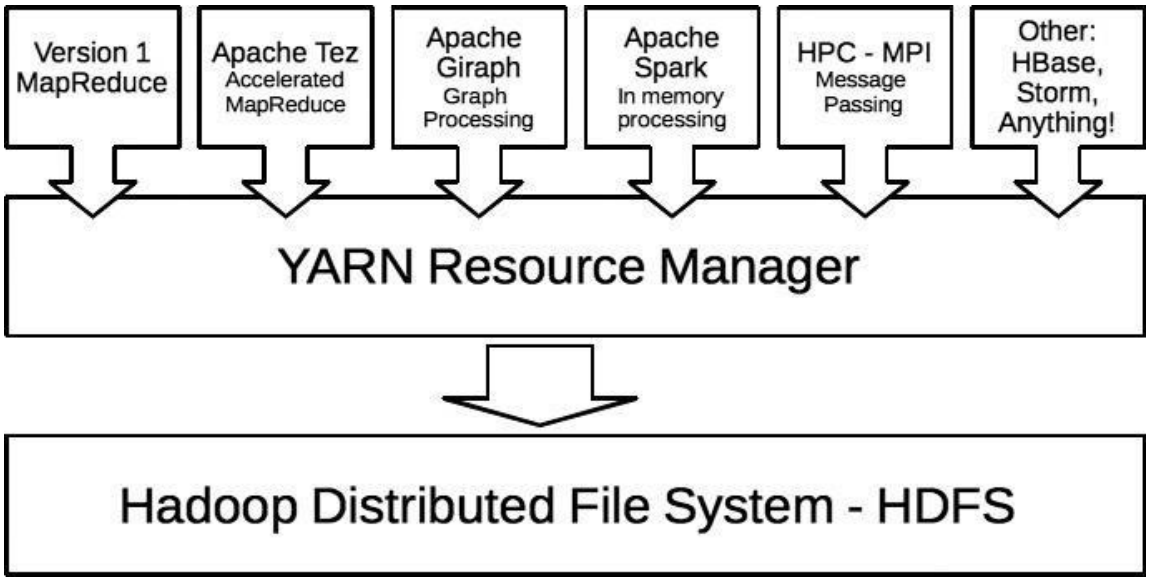


Figure 8.2 Example of the Hadoop version 2 ecosystem. Hadoop version 1 supports batch MapReduce applications only.

This section presents a brief survey of emerging open source YARN application frameworks that are being developed to run under YARN. As of this writing, many YARN frameworks are under active development and the framework landscape is expected to change rapidly. Commercial vendors are also taking advantage of the YARN platform. Consult the webpage for each individual framework for full details of its current stage of development and deployment.

Distributed-Shell

As described earlier in this chapter, Distributed-Shell is an example application included with the Hadoop core components that demonstrates how to write applications on top of YARN. It provides a simple method for running shell commands and scripts in containers in parallel on a Hadoop YARN cluster.

Hadoop MapReduce

MapReduce was the first YARN framework and drove many of YARN's requirements. It is integrated tightly with the rest of the Hadoop ecosystem projects, such as Apache Pig, Apache Hive, and Apache Oozie.

Apache Tez

One great example of a new YARN framework is Apache Tez. Many Hadoop jobs involve the execution of a complex directed acyclic graph (DAG) of tasks using separate MapReduce

stages. Apache Tez generalizes this process and enables these tasks to be spread across stages so that they can be run as a single, all-encompassing job.

Tez can be used as a MapReduce replacement for projects such as Apache Hive and Apache Pig. No changes are needed to the Hive or Pig applications.

Apache Giraph

Apache Giraph is an iterative graph processing system built for high scalability. Facebook, Twitter, and LinkedIn use it to create social graphs of users. Giraph was originally written to run on standard Hadoop V1 using the MapReduce framework, but that approach proved inefficient and totally unnatural for various reasons. The native Giraph implementation under YARN provides the user with an iterative processing model that is not directly available with MapReduce. Support for YARN has been present in Giraph since its own version 1.0 release. In addition, using the flexibility of YARN, the Giraph developers plan on implementing their own web interface to monitor job progress

Hoya: HBase on YARN

The Hoya project creates dynamic and elastic Apache HBase clusters on top of YARN. A client application creates the persistent configuration files, sets up the HBase cluster XML files, and then asks YARN to create an ApplicationMaster. YARN copies all files listed in the client's application-launch request from HDFS into the local file system of the chosen server, and then executes the command to start the Hoya ApplicationMaster. Hoya also asks YARN for the number of containers matching the number of HBase region servers it needs.

Dryad on YARN

Similar to Apache Tez, Microsoft's Dryad provides a DAG as the abstraction of execution flow. This framework is ported to run natively on YARN and is fully compatible with its non-YARN version. The code is written completely in native C++ and C# for worker nodes and uses a thin layer of Java within the application.

Apache Spark

Spark was initially developed for applications in which keeping data in memory improves performance, such as iterative algorithms, which are common in machine learning, and interactive data mining. Spark differs from classic MapReduce in two important ways. First, Spark holds intermediate results in memory, rather than writing them to disk. Second, Spark

supports more than just MapReduce functions; that is, it greatly expands the set of possible analyses that can be executed over HDFS data stores. It also provides APIs in Scala, Java, and Python.

Since 2013, Spark has been running on production YARN clusters at Yahoo!. The advantage of porting and running Spark on top of YARN is the common resource management and a single underlying file system.

Apache Storm

Traditional MapReduce jobs are expected to eventually finish, but Apache Storm continuously processes messages until it is stopped. This framework is designed to process unbounded streams of data in real time. It can be used in any programming language. The basic Storm use-cases include real-time analytics, online machine learning, continuous computation, distributed RPC (remote procedure calls), ETL (extract, transform, and load), and more. Storm provides fast performance, is scalable, is fault tolerant, and provides processing guarantees. It works directly under YARN and takes advantage of the common data and resource management substrate.

Apache REEF: Retainable Evaluator Execution Framework

YARN's flexibility sometimes requires significant effort on the part of application implementers. The steps involved in writing a custom application on YARN include building your own ApplicationMaster, performing client and container management, and handling aspects of fault tolerance, execution flow, coordination, and other concerns. The REEF project by Microsoft recognizes this challenge and factors out several components that are common to many applications, such as storage management, data caching, fault detection, and checkpoints. Framework designers can build their applications on top of REEF more easily than they can build those same applications directly on YARN, and can reuse these common services/libraries. REEF's design makes it suitable for both MapReduce and DAG- like executions as well as iterative and interactive computations.

Hamster: Hadoop and MPI on the Same Cluster

The Message Passing Interface (MPI) is widely used in high-performance computing (HPC). MPI is primarily a set of optimized message-passing library calls for C, C++, and Fortran that operate over popular server interconnects such as Ethernet and InfiniBand. Because users have full control over their YARN containers, there is no reason why MPI applications cannot run within a Hadoop cluster. The Hamster effort is a work-in-progress that provides a good discussion of the issues involved in mapping MPI to a YARN cluster.

Apache Flink: Scalable Batch and Stream Data Processing

Apache Flink is a platform for efficient, distributed, general-purpose data processing. It features powerful programming abstractions in Java and Scala, a high-performance run time, and automatic program optimization. It also offers native support for iterations, incremental iterations, and programs consisting of large DAGs of operations.

Flink is primarily a stream-processing framework that can look like a batch-processing environment. The immediate benefit from this approach is the ability to use the same algorithms for both streaming and batch modes (exactly as is done in Apache Spark). However, Flink can provide low-latency similar to that found in Apache Storm, but which is not available in Apache Spark.

In addition, Flink has its own memory management system, separate from Java's garbage collector. By managing memory explicitly, Flink almost eliminates the memory spikes often seen on Spark clusters.

Apache Slider: Dynamic Application Management

Apache Slider (incubating) is a YARN application to deploy existing distributed applications on YARN, monitor them, and make them larger or smaller as desired in real time.

Applications can be stopped and then started; the distribution of the deployed application across the YARN cluster is persistent and allows for best-effort placement close to the previous locations. Applications that remember the previous placement of data (such as HBase) can exhibit fast startup times by capitalizing on this feature.

YARN monitors the health of "YARN containers" that are hosting parts of the deployed applications. If a container fails, the Slider manager is notified. Slider then requests a new replacement container from the YARN ResourceManager. Some of Slider's other features include user creation of on-demand applications, the ability to stop and restart applications as needed (preemption), and the ability to expand or reduce the number of application containers as needed. The Slider tool is a Java command-line application.

MODULE 3

Chapter 1: Business Intelligence Concepts and Applications

Business intelligence (BI) is an umbrella term that includes a variety of IT applications that are used to analyze an organization's data and communicate the information to relevant users.

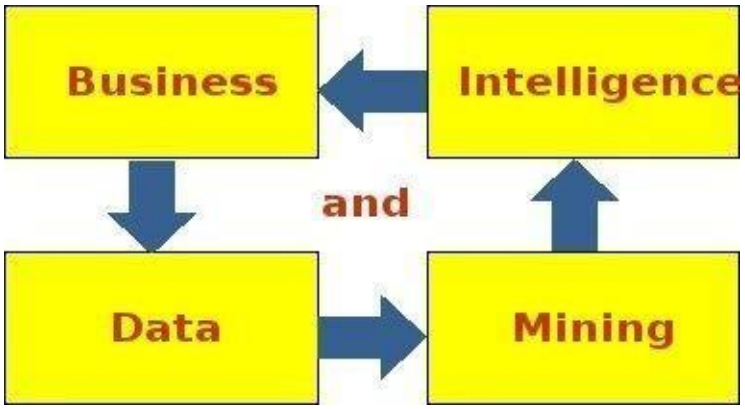


Figure 2.1: BIDM cycle

The nature of life and businesses is to grow. Information is the life-blood of business. Businesses use many techniques for understanding their environment and predicting the future for their own benefit and growth. Decisions are made from facts and feelings. Data- based decisions are more effective than those based on feelings alone. Actions based on accurate data, information, knowledge, experimentation, and testing, using fresh insights, can more likely succeed and lead to sustained growth. One's own data can be the most effective teacher. Therefore, organizations should gather data, sift through it, analyze and mine it, find insights, and then embed those insights into their operating procedures.

1.1 BI for better decisions

The future is inherently uncertain. Risk is the result of a probabilistic world where there are no certainties and complexities abound. People use crystal balls, astrology, palmistry, ground hogs, and also mathematics and numbers to mitigate risk in decision-making. The goal is to make effective decisions, while reducing risk. Businesses calculate risks and make decisions based on a broad set of facts and insights. Reliable knowledge about the future can help managers make the right decisions with lower levels of risk.

The speed of action has risen exponentially with the growth of the Internet. In a hypercompetitive world, the speed of a decision and the consequent action can be a key

advantage. The Internet and mobile technologies allow decisions to be made anytime, anywhere. Ignoring fast-moving changes can threaten the organization's future. Research has shown that an unfavorable comment about the company and its products on social media should not go unaddressed for long. Banks have had to pay huge penalties to Consumer Financial Protection Bureau (CFPB) in United States in 2013 for complaints made on CFPB's websites. On the other hand, a positive sentiment expressed on social media should also be utilized as a potential sales and promotion opportunity, while the opportunity lasts.

1.2 Decision types

There are two main kinds of decisions: strategic decisions and operational decisions. BI can help make both better. Strategic decisions are those that impact the direction of the company. The decision to reach out to a new customer set would be a strategic decision. Operational decisions are more routine and tactical decisions, focused on developing greater efficiency. Updating an old website with new features will be an operational decision.

In strategic decision-making, the goal itself may or may not be clear, and the same is true for the path to reach the goal. The consequences of the decision would be apparent some time later. Thus, one is constantly scanning for new possibilities and new paths to achieve the goals. BI can help with what-if analysis of many possible scenarios. BI can also help create new ideas based on new patterns found from data mining.

Operational decisions can be made more efficient using an analysis of past data. A classification system can be created and modeled using the data of past instances to develop a good model of the domain. This model can help improve operational decisions in the future. BI can help automate operations level decision-making and improve efficiency by making millions of microlevel operational decisions in a model-driven way. For example, a bank might want to make decisions about making financial loans in a more scientific way using data-based models. A decision-tree-based model could provide a consistently accurate loan decisions. Developing such decision tree models is one of the main applications of data mining techniques.

Effective BI has an evolutionary component, as business models evolve. When people and organizations act, new facts (data) are generated. Current business models can be tested against the new data, and it is possible that those models will not hold up well. In that case,

decision models should be revised and new insights should be incorporated. An unending process of generating fresh new insights in real time can help make better decisions, and thus can be a significant competitive advantage.

1.3 BI Tools

BI includes a variety of software tools and techniques to provide the managers with the information and insights needed to run the business. Information can be provided about the current state of affairs with the capability to drill down into details, and also insights about emerging patterns which lead to projections into the future. BI tools include data warehousing, online analytical processing, social media analytics, reporting, dashboards, querying, and data mining.

BI tools can range from very simple tools that could be considered end-user tools, to very sophisticated tools that offer a very broad and complex set of functionality. Thus, Even executives can be their own BI experts, or they can rely on BI specialists to set up the BI mechanisms for them. Thus, large organizations invest in expensive sophisticated BIsolutions that provide good information in real time.

A spreadsheet tool, such as Microsoft Excel, can act as an easy but effective BI tool by itself. Data can be downloaded and stored in the spreadsheet, then analyzed to produce insights, then presented in the form of graphs and tables. This system offers limited automation using macros and other features. The analytical features include basic statistical and financial functions. Pivot tables help do sophisticated what-if analysis. Add-on modules can be installed to enable moderately sophisticated statistical analysis.

A dashboarding system, such as IBM Cognos or Tableau, can offer a sophisticated set of tools for gathering, analyzing, and presenting data. At the user end, modular dashboards can be designed and redesigned easily with a graphical user interface. The back-end data analytical capabilities include many statistical functions. The dashboards are linked to data warehouses at the back end to ensure that the tables and graphs and other elements of the dashboard are updated in real time.

1.4 BI Skills

As data grows and exceeds our capacity to make sense of it, the tools need to evolve, and so should the imagination of the BI specialist. “Data Scientist” has been called as the hottest job

of this decade.

A skilled and experienced BI specialist should be open enough to go outside the box, open the aperture and see a wider perspective that includes more dimensions and variables, in order to find important patterns and insights. The problem needs to be looked at from a wider perspective to consider many more angles that may not be immediately obvious. An imaginative solution should be proposed for the problem so that interesting and useful results can emerge.

A good data mining project begins with an interesting problem to solve. Selecting the right data mining problem is an important skill. The problem should be valuable enough that solving it would be worth the time and expense. It takes a lot of time and energy to gather, organize, cleanse, and prepare the data for mining and other analysis. The data miner needs to persist with the exploration of patterns in the data. The skill level has to be deep enough to engage with the data and make it yield new usefulinsights.

1.5 BI Applications

BI tools are required in almost all industries and functions. The nature of the information and the speed of action may be different across businesses, but every manager today needs access to BI tools to have up-to-date metrics about business performance. Businesses need to embed new insights into their operating processes to ensure that their activities continue to evolve with more efficient practices. The following are some areas of applications of BI and data mining.

1.5.1 Customer Relationship Management

A business exists to serve a customer. A happy customer becomes a repeat customer. A business should understand the needs and sentiments of the customer, sell more of its offerings to the existing customers, and also, expand the pool of customers it serves. BI applications can impact many aspects of marketing.

1. *Maximize the return on marketing campaigns*: Understanding the customer's pain points from data-based analysis can ensure that the marketing messages are fine-tuned to better resonate with customers.

2. *Improve customer retention (churn analysis)*: It is more difficult and expensive to win new customers than it is to retain existing customers. Scoring each customer on their likelihood to quit, can help the business design effective interventions, such as discounts or free services, to retain profitable customers in a cost-effective manner.

3. *Maximize customer value (cross-, up-selling)*: Every contact with the customer should be seen as an opportunity to gauge their current needs. Offering a customer new products and solutions based on those imputed needs can help increase revenue per customer. Even a customer complaint can be seen as an opportunity to wow the customer. Using the knowledge of the customer's history and value, the business can choose to sell a premium service to the customer.

4. *Identify and delight highly-valued customers*. By segmenting the customers, the best customers can be identified. They can be proactively contacted, and delighted, with greater attention and better service. Loyalty programs can be managed more effectively.

5. *Manage brand image*. A business can create a listening post to listen to social media chatter about itself. It can then do sentiment analysis of the text to understand the nature of comments, and respond appropriately to the prospects and customers.

1.5.2 Healthcare and Wellness

Health care is one of the biggest sectors in advanced economies. Evidence- based medicine is the newest trend in data-based health care management. BI applications can help apply the most effective diagnoses and prescriptions for various ailments. They can also help manage public health issues, and reduce waste and fraud.

1. *Diagnose disease in patients*: Diagnosing the cause of a medical condition is the critical first step in a medical engagement. Accurately diagnosing cases of cancer or diabetes can be a matter of life and death for the patient. In addition to the patient's own current situation, many other factors can be considered, including the patient's health history, medication history, family's history, and other environmental factors. This makes diagnosis as much of an art form as it is science. Systems, such as IBM Watson, absorb all the medical research to date and make probabilistic diagnoses in the form of a decision tree, along with a full explanation for their recommendations. These systems take away most of the guess work done by doctors in diagnosing ailments.

2. *Treatment effectiveness*: The prescription of medication and treatment is also a difficult choice out of so many possibilities. For example, there are more than 100 medications for hypertension (high blood pressure) alone. There are also interactions in terms of which drugs work well with others and which drugs do not. Decision trees can help

doctors learn about and prescribe more effective treatments. Thus, the patients could recover their health faster with a lower risk of complications and cost.

3. *Wellness management*: This includes keeping track of patient health records, analyzing customer health trends and proactively advising them to take any needed precautions.

4. *Manage fraud and abuse*: Some medical practitioners have unfortunately been found to conduct unnecessary tests, and/or overbill the government and health insurance companies. Exception reporting systems can identify such providers and action can be taken against them.

5. *Public health management*: The management of public health is one of the important responsibilities of any government. By using effective forecasting tools and techniques, governments can better predict the onset of disease in certain areas in real time. They can thus be better prepared to fight the diseases. Google has been known to predict the movement of certain diseases by tracking the search terms (like flu, vaccine) used in different parts of the world.

1.5.3 Education

As higher education becomes more expensive and competitive, it becomes a great user of data-based decision-making. There is a strong need for efficiency, increasing revenue, and improving the quality of student experience at all levels of education.

1. *Student Enrollment (Recruitment and Retention)*: Marketing to new potential students requires schools to develop profiles of the students that are most likely to attend. Schools can develop models of what kinds of students are attracted to the school, and then reach out to those students. The students at risk of not returning can be flagged, and corrective measures can be taken in time.

2. *Course offerings*: Schools can use the class enrolment data to develop models of which new courses are likely to be more popular with students. This can help increase class size, reduce costs, and improve student satisfaction.

3. *Fund-raising from Alumni and other donors*: Schools can develop predictive models of which alumni are most likely to pledge financial support to the school. Schools can

create a profile for alumni morelikely to pledge donations to the school. This could lead to a reduction in the cost of mailings and other forms of outreach to alumni.

1.5.4 Retail

Retail organizations grow by meeting customer needs with quality products, in a convenient, timely, and cost-effective manner. Understanding emerging

customer shopping patterns can help retailers organize their products, inventory, store layout, and web presence in order to delight their customers, which in turn would help increase revenue and profits. Retailers generate a lot of transaction and logistics data that can be used to diagnose and solve problems.

1. *Optimize inventory levels at different locations*: Retailers need to manage their inventories carefully. Carrying too much inventory imposes carrying costs, while carrying too little inventory can cause stock-outs and lost sales opportunities. Predicting sales trends dynamically can help retailers move inventory to where it is most in demand. Retail organizations can provide their suppliers with real time information about sales of their items, so the suppliers can deliver their product to the right locations
   and minimize stock-outs.

2. *Improve store layout and sales promotions*: A market basket analysis can develop predictive models of which products sell together often. This knowledge of affinities between products can help retailers co-locate those products. Alternatively, those affinity products could be located farther apart to make the customer walk the length and breadth of the store, and thus be exposed to other products. Promotional discounted
   product bundles can be created to push a nonselling item along with a set of products that sell well together.

3. *Optimize logistics for seasonal effects*: Seasonal products offer tremendously profitable short-term sales opportunities, yet they also offer the risk of unsold inventories at the end of the season. Understanding which products are in season in which market can help retailers dynamically manage prices to ensure their inventory is sold during the season. If it is raining in a certain area, then the inventory of umbrella and ponchos could be rapidly moved there from nonrainy areas to help increase sales.

4. *Minimize losses due to limited shelf life*: Perishable goods offer challenges in terms of disposing off the inventory in time. By tracking sales trends, the perishable products at

risk of not selling before the sell- by date, can be suitably discounted and promoted.

1.5.5 Banking

Banks make loans and offer credit cards to millions of customers. They are most interested in improving the quality of loans and reducing bad debts. They also want to retain more good customers, and sell more services to them.

1. *Automate the loan application process*: Decision models can be generated from past data that predict the likelihood of a loan proving successful. These can be inserted in business processes to automate the financial loan approval process.

2. *Detect fraudulent transactions*: Billions of financial transactions happen around the world every day. Exception-seeking models can identify patterns of fraudulent transactions. For example, if money is being transferred to an unrelated account for the first time, it could be a fraudulent transaction.

3. *Maximize customer value (cross-, up-selling).* Selling more products and services to existing customers is often the easiest way to increase revenue. A checking account customer in good standing could be offered home, auto, or educational loans on more favorable terms than other customers, and thus, the value generated from that customer could be increased.

4. *Optimize cash reserves with forecasting*. Banks have to maintain certain liquidity to meet the needs of depositors who may like to withdraw money. Using past data and trend analysis, banks can forecast how much to keep and invest the rest to earn interest.

1.5.6 Financial Services

Stock brokerages are an intensive user of BI systems. Fortunes can be made or lost based on access to accurate and timely information.

1. *Predict changes in bond and stock prices*: Forecasting the price ofstocks and bonds is a favorite pastime of financial experts as well as lay people. Stock transaction data from the past, along with other variables, can be used to predict future price patterns. This can help traders develop long- term trading strategies.

2. *Assess the effect of events on market movements*. Decision models using decision trees can be created to assess the impact of events on changes in market volume and prices. Monetary policy changes (such as Federal Reserve interest rate change) or geopolitical changes (such as war in a part of the world) can be factored into the predictive model to help take action with greater confidence and less risk.

3. *Identify and prevent fraudulent activities in trading*: There have unfortunately been many cases of insider trading, leading to many prominent financial industry stalwarts going to jail. Fraud detection models seek out-of-the-ordinary activities, and help identify and flag fraudulent activity patterns.

1.5.7 Insurance

This industry is a prolific user of prediction models in pricing insurance proposals and managing losses from claims against insured assets.

1. *Forecast claim costs for better business planning*: When natural disasters, such as hurricanes and earthquakes strike, loss of life and property occurs. By using the best available data to model the likelihood (or risk) of such events happening, the insurer can plan for losses and manage resources and profits effectively.

2. *Determine optimal rate plans*: Pricing an insurance rate plan requires covering the potential losses and making a profit. Insurers use actuary tables to project life spans and disease tables to project mortality rates, and thus price themselves competitively yet profitably.

3. *Optimize marketing to specific customers*: By micro-segmenting potential customers, a data-savvy insurer can cherry pick the best customers and leave the less profitable customers to its competitors. Progressive Insurance is a US-based company that is known to actively use data mining to cherry pick customers and increase its profitability.

4. *Identify and prevent fraudulent claim activities*. Patterns can be identified as to where and what kinds of fraud are more likely to occur. Decision-tree-based models can be used to identify and flag fraudulent

   Claims.

1.5.8 Manufacturing

Manufacturing operations are complex systems with inter-related sub- systems. From machines working right, to workers having the right skills, to the right components arriving with the right quality at the right time, to money to source the components, many things have to go right. Toyota's famous lean manufacturing company works on just-in-time inventory systems to optimize investments in inventory and to improve flexibility in their product-mix.

1. *Discover novel patterns to improve product quality*: Quality of a product can also be tracked, and this data can be used to create a predictive model of product quality deteriorating. Many companies, such as automobile companies, have to recall their products if they have found defects that have a public safety implication. Data mining can help with root cause analysis that can be used to identify sources of errors and help improve product quality in the future.

2. *Predict/prevent machinery failures*: Statistically, all equipment is likely to break down at some point in time. Predicting which machine is likely to shut down is a complex process. Decision models to forecast machinery failures could be constructed using past data. Preventive maintenance can be planned, and manufacturing capacity can be adjusted, to account for such maintenance activities.

1.5.9 Telecom

BI in telecom can help with the customer side as well as network side of the operations. Key BI applications include churn management, marketing/customer profiling, network failure, and fraud detection.

1. *Churn management*: Telecom customers have shown a tendency to switch their providers in search for better deals. Telecom companies tend to respond with many incentives and discounts to hold on to customers. However, they need to determine which customers are at a real risk of switching and which others are just negotiating for a better deal. The level of risk should to be factored into the kind of deals and discounts that should be given. Millions of such customer calls happen every month. The telecom
companies need to provide a consistent and data- based way to predict the risk of the customer switching, and then make
an operational decision in real time while the customer call is taking place. A decision-

tree- or a neural network-based system can be used to guide the customer-service call operator to make the right decisions for the company, in a consistent manner.

2. *Marketing and product creation*. In addition to customer data, telecom companies also store call detail records (CDRs), which can be analyzed to precisely describe the calling behavior of each customer. This unique data can be used to profile customers and then can be used for creating new products/services bundles for marketing purposes. An American telecom company, MCI, created a program called Friends & Family that allowed free calls with one's friends and family on that network, and thus, effectively locked many people into their network.

3. *Network failure management*: Failure of telecom networks for technical failures or malicious attacks can have devastating impacts on people, businesses, and society. In telecom infrastructure, some equipment will likely fail with certain mean time between failures. Modeling the failure pattern of various components of the network can help with preventive maintenance and capacity planning.

4. *Fraud Management*: There are many kinds of fraud in consumer transactions. Subscription fraud occurs when a customer opens an account with the intention of never paying for the services. Superimposition fraud involves illegitimate activity by a person other than the legitimate account holder. Decision rules can be developed to analyze each CDR in real time to identify chances of fraud and take effective action.

1.5.10 Public Sector

Government gathers a large amount of data by virtue of their regulatory function. That data could be analyzed for developing models of effective functioning. There are innumerable applications that can benefit from mining that data. A couple of sample applications are shown here.

1. *Law enforcement*: Social behavior is a lot more patterned and predictable than one would imagine. For example, Los Angeles Police Department (LAPD) mined the data from its 13 million crime records over 80 years and developed models of what kind of crime going to happen when and where. By increasing patrolling in those particular areas, LAPD was able to reduce property crime by 27 percent. Internet chatter can be analyzed to learn of and prevent any evil designs.

2. *Scientific research*: Any large collection of research data is amenable to being mined for patterns and insights. Protein folding (microbiology), nuclear reaction analysis (sub-atomic physics), disease control (public health) are some examples where data mining can yield powerful new insights.

Chapter 2: Data Warehousing

A data warehouse (DW) is an organized collection of integrated, subject- oriented databases designed to support decision support functions. DW is organized at the right level of granularity to provide clean enterprise-wide data in a standardized format for reports, queries, and analysis. DW is physically and functionally separate from an operational and transactional database. Creating a DW for analysis and queries represents significant investment in time and effort. It has to be constantly kept up-to-date for it to be useful. DW offers many business and technical benefits.

DW supports business reporting and data mining activities. It can facilitate distributed access to up-to-date business knowledge for departments and functions, thus improving business efficiency and customer service. DW can present a competitive advantage by facilitating decision making and helping reform business processes.

DW enables a consolidated view of corporate data, all cleaned and organized. Thus, the entire organization can see an integrated view of itself. DW thus provides better and timely information. It simplifies data access and allows end users to perform extensive analysis.

2.1 Design Considerations for DW

The objective of DW is to provide business knowledge to support decision making. For DW to serve its objective, it should be aligned around those decisions. It should be comprehensive, easy to access, and up-to-date. Here are some requirements for a good DW:

1. *Subject oriented*: To be effective, a DW should be designed around a subject domain, i.e. to help solve a certain category ofproblems.

2. *Integrated*: The DW should include data from many functions that can shed light on a particular subject area. Thus the organization can benefit from a comprehensive view of the subject area.

3. *Time-variant (time series):* The data in DW should grow at daily or other chosen intervals. That allows latest comparisons over time.

4. *Nonvolatile*: DW should be persistent, that is, it should not be created on the fly from the operations databases. Thus, DW is consistently available for analysis, across the organization and over time.

5. *Summarized*: DW contains rolled-up data at the right level for queries and analysis. The process of rolling up the data helps create consistent granularity for effective comparisons. It also helps reduces the number of variables or dimensions of the data to make them more meaningful for the decision makers.

6. *Not normalized*: DW often uses a star schema, which is a rectangular central table, surrounded by some look-up tables. The single table view significantly enhances speed of queries.

7. *Metadata*: Many of the variables in the database are computed from other variables in the operational database. For example, total daily sales may be a computed field. The method of its calculation for each variable should be effectively documented. Every element in the DW should be sufficiently well-defined.

8. *Near Real-time and/or right-time (active*): DWs should be updated in near real-time in many high transaction volume industries, such as airlines. The cost of implementing and updating DW in real time could be discouraging though. Another downside of real- time DW is the possibilities of inconsistencies in reports drawn just a few minutes apart.

2.2 DW Development Approaches

There are two fundamentally different approaches to developing DW: top down and bottom up. The top-down approach is to make a comprehensive DW that covers all the reporting needs of the enterprise. The bottom-up approach is to produce small data marts, for the reporting needs of different departments or functions, as needed. The smaller data marts will eventually align to deliver comprehensive EDW capabilities. The top-down approach provides consistency but takes more time and resources. The bottom-up approach leads to healthy local ownership and maintainability of data (Table 3.1).

	Functional Data Mart	**Enterprise Data Warehouse**
Scope	One subject or functional area	Complete enterprise data needs
Value	Functional area reporting and insights	Deeper insights connecting multiple functional areas
Target organization	Decentralized management	Centralized management
Time	Low to medium	High

Cost	Low	High
Size	Small to medium	Medium to large
Approach	Bottom up	Top down
Complexity	Low (fewer data transformations)	High (data standardization)
Technology	Smaller scale servers and databases	Industrial strength

Table 2.1: Comparing Data Mart and Data Warehouse

2.3 DW Architecture

DW has four key elements (Figure 2.1). The first element is the data sources that provide the raw data. The second element is the process of transforming that data to meet the decision needs. The third element is the methods of regularly and accurately loading of that data into EDW or data marts. The fourth element is the data access and analysis part, where devices and applications use the data from DW to deliver insights and other benefits to users.

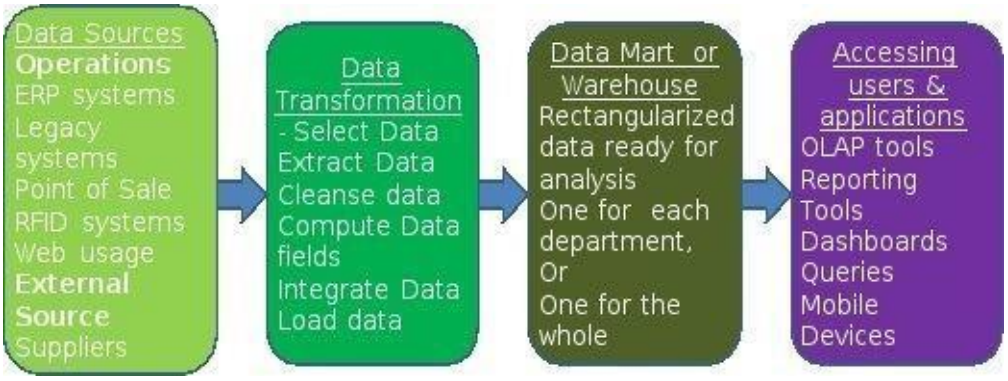


Figure 2.1: Data Warehousing Architecture

2.3.1 Data Sources

Data Warehouses are created from structured data sources. Unstructured data such as text data would need to be structured before inserted into the DW.

1. *Operations data:* This includes data from all business applications, including from ERPs systems that form the backbone of an organization's IT systems. The data to be extracted will depend upon the subject matter of the data warehouse. For example, for a sales/marketing data mart, only the data about customers, orders, customer service, and so on would be extracted.

2. *Specialized applications*: This includes applications such as Point of Sale (POS) terminals, and e-commerce applications, that also provide customer-facing data. Supplier data could come from Supply Chain Management systems. Planning and budget data should also be added as needed for making comparisons against targets.

3. *External syndicated data*: This includes publicly available data such as weather or economic activity data. It could also be added to the DW, as needed, to provide good contextual information to decisionmakers.

2.3.2 Data Loading Processes

The heart of a useful DW is the processes to populate the DW with good quality data. This is called the Extract-Transform-Load (ETL) cycle.

1. Data should be extracted from the operational (transactional) database sources, as well as from other applications, on a regularbasis.
2. The extracted data should be aligned together by key fields and integrated into a single data set. It should be cleansed of any irregularities or missing values. It should be rolled-up together to the same level of granularity. Desired fields, such as daily sales totals, should be computed. The entire data should then be brought to the same format as the central table of DW.
3. This transformed data should then be uploaded into the DW.

This ETL process should be run at a regular frequency. Daily transaction data can be extracted from ERPs, transformed, and uploaded to the database the same night. Thus, the DW is up to date every morning. If a DW is needed for near-real-time information access, then the ETL processes would need to be executed more frequently. ETL work is usually done using automated using programming scripts that are written, tested, and then deployed for periodically updating the DW.

2.4 Data Warehouse Design

Star schema is the preferred data architecture for most DWs. There is a central fact table that provides most of the information of interest. There are lookup tables that provide detailed values for codes used in the central table. For example, the central table may use digits to represent a sales person. The lookup table will help provide the name for that sales person code. Here is an example of a star schema for a data mart for monitoring sales performance (Figure 2.2).

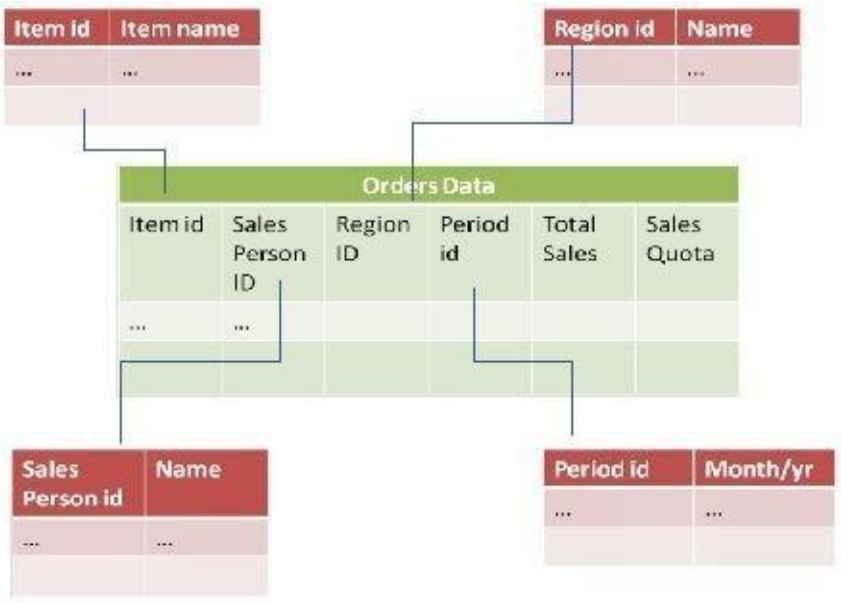


Figure 2.2: Star Schema Architecture for DW

Other schemas include the snowflake architecture. The difference between a star and snowflake is that in the latter, the look-up tables can have their own further look up tables.

There are many technology choices for developing DW. This includes selecting the right database management system and the right set of data management tools. There are a few big and reliable providers of DW systems. The provider of the operational DBMS may be chosen for DW also. Alternatively, a best-of-breed DW vendor could be used. There are also a variety of tools out there for data migration, data upload, data retrieval, and data analysis.

2.5 DW Access

Data from the DW could be accessed for many purposes, by many users, through many devices.

1. A primary use of DW is to produce routine management and monitoring reports. For example, a sales performance report would show sales by many dimensions, and compared with plan. A dashboarding system will use data from the warehouse and present analysis to users. The data from DW can be used to populate customized performance dashboards for executives. The dashboard could include drill-down capabilities to analyze the performance data for root cause analysis.

2. The data from the DW could be used for ad-hoc queries and any other applications that make use of the internal data.

3. Data from DW is used to provide data for mining purposes. Parts of the data would be extracted, and then combined with other relevant data, for data mining.

2.6 DW Best Practices

A data warehousing project reflects a significant investment into information technology (IT). All of the best practices in implementing any IT project should be followed.

1. The DW project should *align with the corporate strategy*. Top management should be consulted for setting objectives. Financial viability (ROI) should be established. The project must be managed by both IT and business professionals. The DW design should be carefully tested before beginning development work. It is often much more expensive to redesign after development work has begun.

2. It is important to *manage user expectations*. The data warehouse should be built incrementally. Users should be trained in using the system so they can absorb the many features of the system.

3. *Quality and adaptability* should be built in from the start. Only relevant, cleansed, and high-quality data should be loaded. The system should be able to adapt to new tools for access. As business needs change, new data marts may need to be created for new needs.

Chapter 3: Data Mining

Data mining is the art and science of discovering knowledge, insights, and patterns in data. It is the act of extracting useful patterns from an organized collection of data. Patterns must be valid, novel, potentially useful, and understandable. The implicit assumption is that data about the past can reveal patterns of activity that can be projected into the future.

Data mining is a multidisciplinary field that borrows techniques from a variety of fields. It utilizes the knowledge of data quality and data organizing from the databases area. It draws modeling and analytical techniques from statistics and computer science (artificial intelligence) areas. It also draws the knowledge of decision-making from the field of business management.

The field of data mining emerged in the context of pattern recognition in defense, such as identifying a friend-or-foe on a battlefield. Like many other defense-inspired technologies, it has evolved to help gain a competitive advantage in business.

For example, "customers who buy cheese and milk also buy bread 90 percent of the time" would be a useful pattern for a grocery store, which can then stock the products appropriately. Similarly, "people with blood pressure greater than 160 and an age greater than 65 were at a high risk of dying from a heart stroke" is of great diagnostic value for doctors, who can then focus on treating such patients with urgent care and great sensitivity.

3.1 Gathering and selecting data

The total amount of data in the world is doubling every 18 months. There is an ever-growing avalanche of data coming with higher velocity, volume, and variety. One has to quickly use it or lose it. Smart data mining requires choosing where to play. One has to make judicious decisions about what to gather and what to ignore, based on the purpose of the data mining exercises. It is like deciding where to fish; as not all streams of data will be equally rich in potential insights.

To learn from data, quality data needs to be effectively gathered, cleaned and organized, and then efficiently mined. One requires the skills and technologies for consolidation and integration of data elements from many sources. Most organizations develop an enterprise data model (EDM) to organize their data. An EDM is a unified, high-level model of all the data stored in an organization's databases. The EDM is usually inclusive of the data generated

from all internal systems. The EDM provides the basic menu of data to create a data warehouse for a particular decision-making purpose. DWs help organize all this data in an easy and usable manner so that it can be selected and deployed for mining. The EDM can also help imagine what relevant external data should be gathered to provide context and develop good predictive relationships with the internal data. In the United States, the various federal and local governments and their regulatory agencies make a vast variety and quantity of data available at data.gov.

Gathering and curating data takes time and effort, particularly when it is unstructured or semistructured. Unstructured data can come in many forms like databases, blogs, images, videos, audio, and chats. There are streams of unstructured social media data from blogs, chats, and tweets. There are streams of machine-generated data from connected machines, RFID tags, the internet of things, and so on. Eventually the data should be *rectangularized*, that is, put in rectangular data shapes with clear columns and rows, before submitting it to data mining.

Knowledge of the business domain helps select the right streams of data for pursuing new insights. Only the data that suits the nature of the problem being solved should be gathered. The data elements should be relevant, and suitably address the problem being solved. They could directly impact the problem, or they could be a suitable proxy for the effect being measured. Select data could also be gathered from the data warehouse. Every industry and function will have its own requirements and constraints. The health care industry will provide a different type of data with different data names. The HR function would provide different kinds of data. There would be different issues of quality and privacy for these data.

3.2 Data cleansing and preparation

The quality of data is critical to the success and value of the data mining project. Otherwise, the situation will be of the kind of garbage in and garbage out (GIGO). The quality of incoming data varies by the source and nature of data. Data from internal operations is likely to be of higher quality, as it will be accurate and consistent. Data from social media and other public sources is less under the control of business, and is less likely to bereliable.

Data almost certainly needs to be cleansed and transformed before it can be used for data mining. There are many ways in what data may need to be cleansed – filling missing values,

reigning in the effects of outliers, transforming fields, binning continuous variables, and much more – before it can be ready for analysis. Data cleansing and preparation is a labor- intensive or semi-automated activity that can take up to 60-70% of the time needed for a data mining project.

1. *Duplicate data needs to be removed*. The same data may be received from multiple sources. When merging the data sets, data must be de- duped.
2. *Missing values need to be filled in*, or those rows should be removed from analysis. Missing values can be filled in with average or modal or default values.
3. *Data elements should be comparable.* They may need to be (a) transformed from one unit to another. For example, total costs of health care and the total number of patients may need to be reduced to cost/patient to allow comparability of that value. Data elements may need to be adjusted to make them (b) comparable over time also. For example, currency values may need to be adjusted for inflation; they would need to be converted to the same base year for comparability. They may need to be converted to a common currency. Data should be

   (c) stored at the same granularity to ensure comparability. For example, sales data may be available daily, but the sales person compensation data may only be available monthly. To relate these variables, the data must be brought to the lowest common denominator, in this case, monthly.
4. *Continuous values may need to be binned* into a few buckets to help with some analyses. For instance, work experience could be binned as low, medium, and high.
5. *Outlier data elements need to be removed* after careful review, to avoid the skewing of results. For example, one big donor could skew the analysis of alumni donors in an educational setting.
6. *Ensure that the data is representative of the phenomena* under analysis by correcting for any biases in the selection of data. For example, if the data includes many more members of one gender than is typical of the population of interest, then adjustments need to be applied to the data.
7. Data may need to be selected to *increase information density*. Some data may not show much variability, because it was not properly recorded or for other reasons. This data may dull the effects of other differences in the data and should be removed to improve the information density of the data.

3.3 Outputs of Data Mining

Data mining techniques can serve different types of objectives. The outputs of data mining will reflect the objective being served. There are many ways of representing the outputs of data mining.

One popular form of data mining output is a decision tree. It is a hierarchically branched structure that helps visually follow the steps to make a model-based decision. The tree may have certain attributes, such as probabilities assigned to each branch. A related format is a set of business rules, which are if-then statements that show causality. A decision tree can be mapped to business rules. If the objective function is prediction, then a decision tree or business rules are the most appropriate mode of representing the output.

The output can be in the form of a regression equation or mathematical function that represents the best fitting curve to represent the data. This equation may include linear and nonlinear terms. Regression equations are a good way of representing the output of classification exercises. These are also a good representation of forecasting formulae.

Population "centroid" is a statistical measure for describing central tendencies of a collection of data points. These might be defined in a multidimensional space. For example, a centroid could be "middle-aged, highly educated, high- net worth professionals, married with two children, living in the coastal areas". Or a population of "20-something, ivy-league-educated, tech entrepreneurs based in Silicon Valley". Or it could be a collection of "vehicles more than 20 years old, giving low mileage per gallon, which failed environmental inspection". These are typical representations of the output of a cluster analysis exercise.

Business rules are an appropriate representation of the output of a market basket analysis exercise. These rules are if-then statements with some probability parameters associated with each rule. For example, those that buy milk and bread will also buy butter (with 80 percent probability).

The output can be in the form of a regression equation or mathematical function that represents the best fitting curve to represent the data. This equation may include linear and non-linear terms. Regression equations are a good way of representing the output of classification exercises. These are also a good representation of forecasting formulae. Population 'centroid' is a statistical measure for describing central tendencies of a collection

of data points. These might be defined in a multi-dimensional space. For example, a centroid could be "middle-aged, highly educated, high- net worth professionals, married with 2 children, living in the coastal areas". Or a population of "20-something, ivy-league-educated, tech entrepreneurs based in Silicon Valley". Or a collection of "vehicles more than 20 years old, giving low mileage per gallon, that failed the environmental inspection". These are typical representations of the output of a cluster analysis exercise.

Business rules are an appropriate representation of the output of a market- basket analysis exercise. These rules are if-then statements with some probability parameters associated with each rule. For example, those that buy milk and bread, will also buy butter (with 80% probability).

3.4 Evaluating Data Mining Results

There are two primary kinds of data mining processes: supervised learning and unsupervised learning. In supervised learning, a decision model can be created using past data, and the model can then be used to predict the correct answer for future data instances. Classification is the main category of supervised learning activity. There are many techniques for classification, decision trees being the most popular one. Each of these techniques can be implemented with many algorithms. A common metric for all of classification techniques is predictive accuracy.

Predictive Accuracy = (Correct Predictions) / Total Predictions

Suppose a data mining project has been initiated to develop a predictive model for cancer patients using a decision tree. Using a relevant set of variables and data instances, a decision tree model has been created. The model is then used to predict other data instances. When a true positive data point is positive, that is a correct prediction, called a true positive (TP). Similarly, when a true negative data point is classified as negative, that is a true negative (TN). On the other hand, when a true-positive data point is classified by the model as negative, that is an incorrect prediction, called a false negative (FN). Similarly, when a true- negative data point is classified as positive, that is classified as a false positive (FP). This is represented using the confusion matrix (Figure 3.1).

ConfusionMatrix		True Class	
		Positive	*Negative*
Predicted Class	Positive	**True Positive (TP)**	False Positive (FP)
Predicted class	Negative	False Negative (FN)	**True Negative (TN)**

Figure 3.1: Confusion Matrix

Thus the predictive accuracy can be specified by the following formula. Predictive Accuracy = (TP +TN) / (TP + TN + FP + FN).

All classification techniques have a predictive accuracy associated with a predictive model. The highest value can be 100%. In practice, predictive models with more than 70% accuracy can be considered usable in business domains, depending upon the nature of the business.

There are no good objective measures to judge the accuracy of unsupervised learning techniques such as Cluster Analysis. There is no single right answer for the results of these techniques. For example, the value of the segmentation model depends upon the value the decision-maker sees in thoseresults.

3.5 Data Mining Techniques

Data may be mined to help make more efficient decisions in the future. Or it may be used to explore the data to find interesting associative patterns. The right technique depends upon the kind of problem being solved (Figure 3.2).

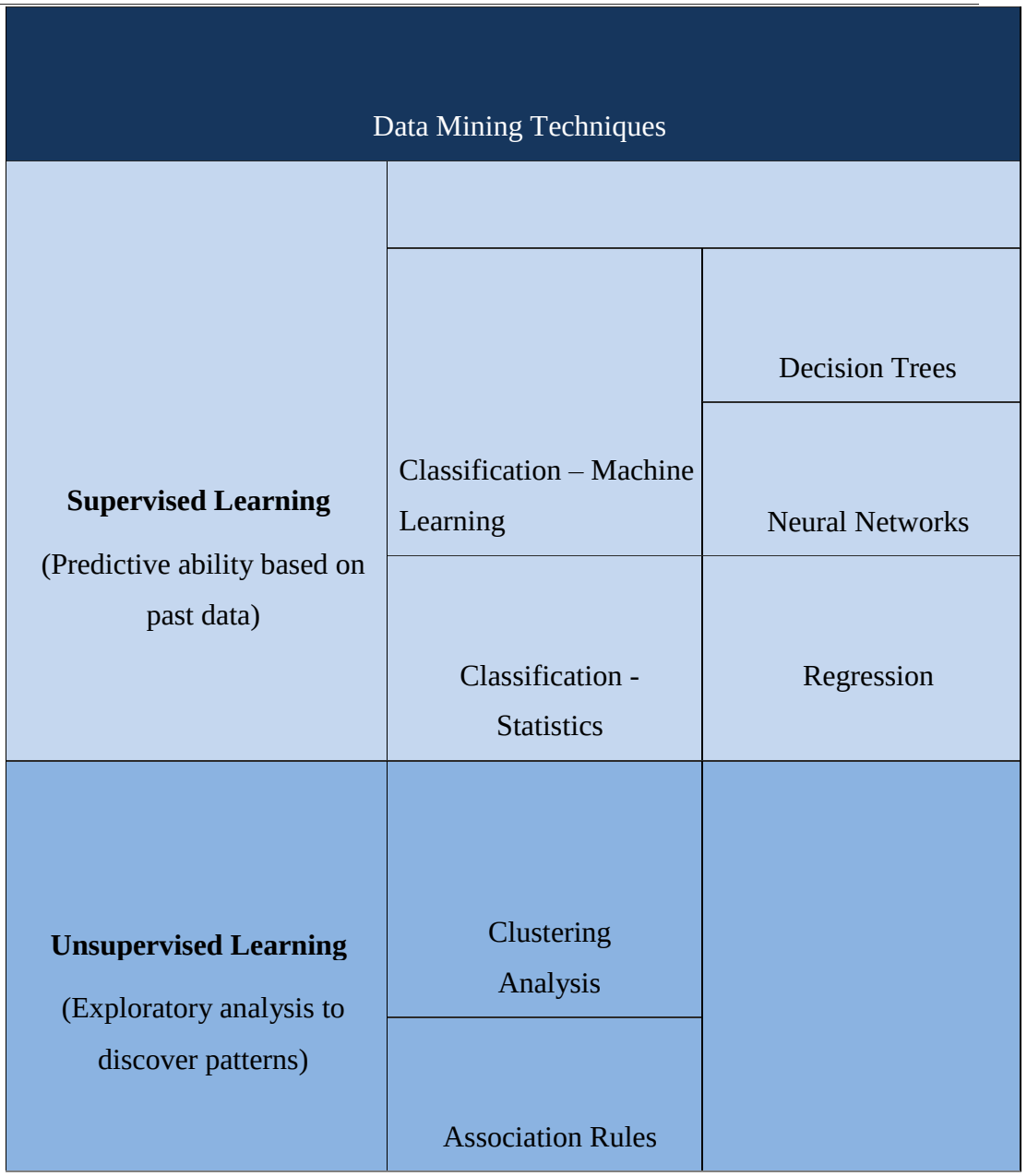


Figure 3.2: Important Data Mining Techniques

The most important class of problems solved using data mining are classification problems. Classification techniques are called supervised learning as there is a way to supervise whether the model is providing the right or wrong answers. These are problems where data from past decisions is mined to extract the few rules and patterns that would improve the accuracy of the decision making process in the future. The data of past decisions is organized and mined for decision rules or equations, that are then codified to produce more accurate decisions.

Decision trees are the most popular data mining technique, for many reasons.

1. Decision trees are easy to understand and easy to use, by analysts as well as executives. They also show a high predictive accuracy.
2. Decision trees select the most relevant variables automatically out of all the available variables for decision making.
3. Decision trees are tolerant of data quality issues and do not require much data preparation from the users.
4. Even non-linear relationships can be handled well by decisiontrees.

There are many algorithms to implement decision trees. Some of the popular ones are C5, CART and CHAID.

Regression is a most popular statistical data mining technique. The goal of regression is to derive a smooth well-defined curve to best the data. Regression analysis techniques, for example, can be used to model and predict the energy consumption as a function of daily temperature. Simply plotting the data may show a non-linear curve. Applying a non-linear regression equation will fit the data very well with high accuracy. Once such a regression model has been developed, the energy consumption on any future day can be predicted using this equation. The accuracy of the regression model depends entirely upon the dataset used and not at all on the algorithm or tools used.

Artificial Neural Networks (ANN) is a sophisticated data mining technique from the Artificial Intelligence stream in Computer Science. It mimics the behavior of human neural structure: Neurons receive stimuli, process them, and communicate their results to other neurons successively, and eventually a neuron outputs a decision. A decision task may be processed by just one neuron and the result may be communicated soon. Alternatively, there could be many layers of neurons involved in a decision task, depending upon the complexity of the domain. The neural network can be trained by making a decision over and over again with many data points. It will continue to learn by adjusting its internal computation and communication parameters based on feedback received on its previous decisions. The intermediate values passed within the layers of neurons may not make any intuitive sense to an observer. Thus, the neural networks are considered a black-box system.

At some point, the neural network will have learned enough and begin to match the predictive accuracy of a human expert or alternative classification techniques. The predictions of some ANNs that have been trained over a long period of time with a large amount of data have become decisively more accurate than human experts. At that point, the ANNs can begin to be seriously considered for deployment, in real situations in real time. ANNs are popular because they are eventually able to reach a high predictive accuracy. ANNs are also relatively simple to implement and do not have any issues with data quality. However, ANNs require a lot of data to train it to develop good predictive ability.

Cluster Analysis is an exploratory learning technique that helps in identifying a set of similar groups in the data. It is a technique used for automatic identification of natural groupings of things. Data instances that are similar to (or near) each other are categorized into one cluster, while data instances that are very different (or far away) from each other are categorized into separate clusters. There can be any number of clusters that could be produced by the data. The K-means technique is a popular technique and allows the user guidance in selecting the right number (K) of clusters from the data.

Clustering is also known as the segmentation technique. It helps divide and conquer large data sets. The technique shows the clusters of things from past data. The output is the centroids for each cluster and the allocation of data points to their cluster. The centroid definition is used to assign new data instances can be assigned to their cluster homes. Clustering is also a part of the artificial intelligence family of techniques.

Association rules are a popular data mining method in business, especially where selling is involved. Also known as market basket analysis, it helps in answering questions about cross-selling opportunities. This is the heart of the personalization engine used by ecommerce sites like Amazon.com and streaming movie sites like Netflix.com. The technique helps find interesting relationships (affinities) between variables (items or events). These are represented as rules of the form $X \rightarrow Y$, where X and Y are sets of data items. A form of unsupervised learning, it has no dependent variable; and there are no right or wrong answers. There are just stronger and weaker affinities. Thus, each rule has a confidence level assigned to it. A part of the machine learning family, this technique achieved legendary status when a fascinating relationship was found in the sales of diapers and beers.

3.6 Tools and Platforms for Data Mining

Data mining tools have existed for many decades. However, they have recently become more important as the values of data have grown and the field of big data analytics has come into prominence. There are a wide range of data mining platforms available in the market today.

1. *Simple or sophisticated*: There are simple end-user data mining tools such as MS Excel, and there are more sophisticated tools such as IBM SPSS Modeler.
2. *Stand-alone or Embedded*: There are stand alone tools and there are tools embedded in an existing transaction processing or data warehousing or ERP system.
3. *Open source or Commercial*: There are open source and freely available tools such as Weka, and there are commercial products.
4. *User interface*: There are text-based tools that require some programming skills, and there are GUI-based drag-and-drop format tools.
5. *Data formats*: There are tools that work only on proprietary data formats and there are those directly accept data from a host of popular data management tools formats.

Here we compare three platforms that we have used extensively and effectively for many data mining projects.

Table 3.1: Comparison of Popular Data Mining Platforms

Feature	**Excel**	**IBM SPSS Modeler**	**Weka**
Ownership	Commercial	Commercial, expensive	Open-source, free
Data Mining Features	Limited; extensible with add-on modules	Extensive features, unlimited data sizes	Extensive, performance issues with large data

Stand-alone	Stand-alone	Embedded in BI software suites	Stand-alone
User skills needed	End-users	For skilled BI analysts	Skilled BI analysts
User interface	Text and click, Easy	Drag & Drop use, colorful, beautiful GUI	GUI, mostly b&w text output
Data formats	Industry-standard	Variety of data sources accepted	Proprietary

MS Excel is a relatively simple and easy data mining tool. It can get quite versatile once Analyst Pack and some other add-on products are installed on it.

IBM's SPSS Modeler is an industry-leading data mining platform. If offers a powerful set of tools and algorithms for most popular data mining capabilities. It has colorful GUI format with drag-and-drop capabilities. Itcan accept data in multiple formats including reading Excel filesdirectly.

Weka is an open-source GUI based tool that offers a large number of data mining algorithms. ERP systems include some data analytic capabilities, too. SAP has its Business Objects (BO) software. BO is considered one of the leading BI suites in the industry, and is often used by organizations that use SAP.

3.7 Data Mining Best Practices

Effective and successful use of data mining activity requires both business and technology skills. The business aspects help understand the domain and the key questions. It also helps one imagine possible relationships in the data, and create hypotheses to test it. The IT aspects help

fetch the data from many sources, clean up the data, assemble it to meet the needs of the business problem, and then run the data mining techniques on the platform.

An important element is to go after the problem iteratively. It is better to divide and conquer the problem with smaller amounts of data, and get closer to the heart of the solution in an iterative sequence of steps. There are several best practices learned from the use of data mining techniques over a long period of time. The Data Mining industry has proposed a Cross-Industry Standard Process for Data Mining (CRISP-DM). It has six essential steps (Figure 4.3):

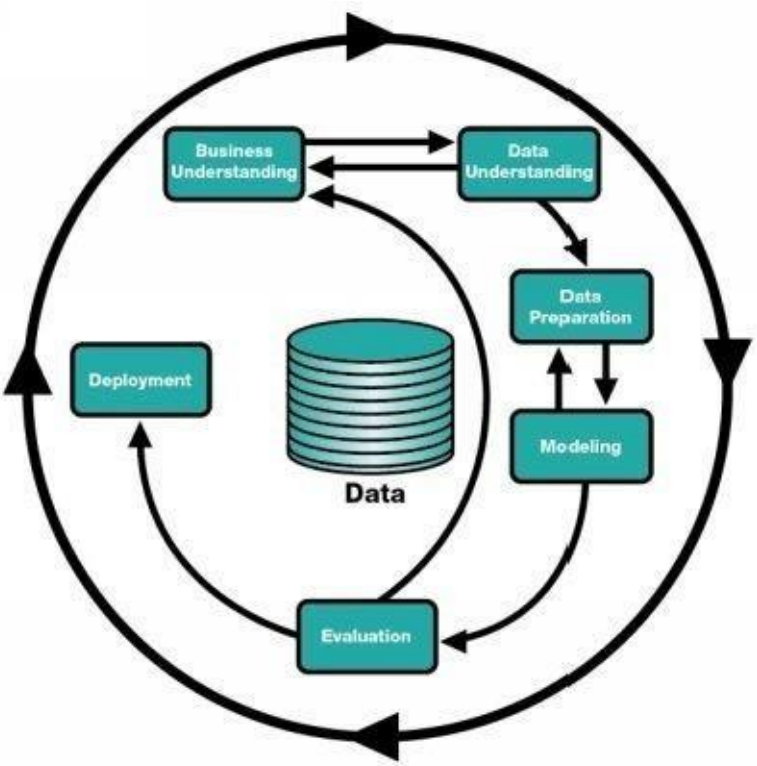


Figure 3.3: CRISP-DM Data Mining cycle

1. *Business Understanding*: The first and most important step in data mining is asking the right business questions. A question is a good one if answering it would lead to large payoffs for the organization, financially and otherwise. In other words, selecting a data mining project is like any other project, in that it should show strong payoffs if the project is successful. There should be strong executive support for the data mining project, which means that the project aligns well with the business strategy. A related
important step is to be creative and open in proposing imaginative hypotheses for the solution. Thinking outside the box is important, both in terms of a proposed model as well in the data sets available and required.

2. *Data Understanding*: A related important step is to understand the data available for mining. One needs to be imaginative in scouring for many elements of data through many sources in helping address the hypotheses to solve a problem. Without relevant data, the hypotheses cannot be tested.
3. *Data Preparation*: The data should be relevant, clean and of high quality. It's important to assemble a team that has a mix of technical and business skills, who understand the domain and the data. Data cleaning can take 60-70% of the time in a data mining project. It may be desirable to continue to experiment and add new data elements from externalsources of data that could help improve predictive accuracy.
4. *Modeling*: This is the actual task of running many algorithms using the available data to discover if the hypotheses are supported. Patience is required in continuously engaging with the data until the data yields some good insights. A host of modeling tools and algorithms should be used. A tool could be tried with different options, such as running different decision tree algorithms.
5. *Model Evaluation*: One should not accept what the data says at first. It is better to triangulate the analysis by applying multiple data mining techniques, and conducting many what-if scenarios, to build confidence in the solution. One should evaluate and improve the model's predictive accuracy with more test data. When the accuracy has reached some satisfactory level, then the model should be deployed.
6. *Dissemination and rollout:* It is important that the data mining solution is presented to the key stakeholders, and is deployed in the organization. Otherwise the project will be a waste of time and will be a setback for establishing and supporting a data-based decision-process culture in the organization. The model should be eventually embedded in the organization's business processes.

3.8 Myths about data mining

There are many myths about this area, scaring away many business executives from using data mining. Data Mining is a mindset that presupposes a faith in the ability to reveal insights. By itself, data mining is not too hard, nor is it too easy. It does require a disciplined approach and some cross- disciplinary skills.

Myth #1: Data Mining is about algorithms. Data mining is used by business to answer important and practical business questions. Formulating the problem statement correctly and identifying imaginative solutions for testing are far more important before the data mining

algorithms gets called in. Understanding the relative strengths of various algorithms is helpful but not mandatory.

Myth #2: Data Mining is about predictive accuracy. While important, predictive accuracy is a feature of the algorithm. As in myth#1, the quality of output is a strong function of the right problem, right hypothesis, and the right data.

Myth #3: Data Mining requires a data warehouse. While the presence of a data warehouse assists in the gathering of information, sometimes the creation of the data warehouse itself can benefit from some exploratory data mining. Some data mining problems may benefit from clean data available directly from the DW, but a DW is not mandatory.

Myth #4: Data Mining requires large quantities of data. Many interesting data mining exercises are done using small or medium sized data sets, at low costs, using end-user tools.

Myth #5: Data Mining requires a technology expert. Many interesting data mining exercises are done by end-users and executives using simpleeveryday tools like spreadsheets.

3.9 Data Mining Mistakes

Data mining is an exercise in extracting non-trivial useful patterns in the data. It requires a lot of preparation and patience to pursue the many leads that data may provide. Much domain knowledge, tools and skill is required to find such patterns. Here are some of the more common mistakes in doing data mining, and should be avoided.

Mistake #1: *Selecting the wrong problem for data mining*: Without the right goals or having no goals, data mining leads to a waste of time. Getting the right answer to an irrelevant question could be interesting, but it would be pointless from a business perspective. A good goal would be one that would deliver a good ROI to the organization.

Mistake #2: Buried under mountains of data without clear metadata: It is more important to be engaged with the data, than to have lots of data. The relevant data required may be much less than initially thought. There may be insufficient knowledge about the data, or metadata. Examine the data with a critical eye and do not naively believe everything you are told about the data.

Mistake #3: Disorganized data mining: Without clear goals, much time is wasted. Doing the same tests using the same mining algorithms repeatedly and blindly, without thinking about the next stage, without a plan, would lead to wasted time and energy. This can come from being sloppy about keeping track of the data mining procedure and results. Not leaving sufficient time for data acquisition, selection and preparation can lead to data quality issues, and GIGO. Similarly not providing enough time for testing the model, training the users and deploying the system can make the project a failure.

Mistake #4: Insufficient business knowledge: Without a deep understanding of the business domain, the results would be gibberish and meaningless. Don't make erroneous assumptions, courtesy of experts. Don't rule out anything when observing data analysis results. Don't ignore suspicious (good or bad) findings and quickly move on. Be open to surprises. Even when insights emerge at one level, it is important to slice and dice the data at other levels to see if more powerful insights can be extracted.

Mistake #5: Incompatibility of data mining tools and datasets. All the tools from data gathering, preparation, mining, and visualization, should work together. Use tools that can work with data from multiple sources in multiple industry standard formats.

Mistake #6: Looking only at aggregated results and not at individual records/predictions. It is possible that the right results at the aggregate level provide absurd conclusions at an individual record level. Diving into the data at the right angle can yield insights at many levels of data.

Mistake #7: Not measuring your results differently from the way your sponsor measures them. If the data mining team loses its sense of business objectives, and beginning to mine data for its own sake, it will lose respect and executive support very quickly.

Chapter 4: Data Visualization

Data Visualization is the art and science of making data easy to understand and consume, for the end user. Ideal visualization shows the right amount of data, in the right order, in the right visual form, to convey the high priority information. The right visualization requires an understanding of the consumer's needs, nature of the data, and the many tools and techniques available to present data. The right visualization arises from a complete understanding of the totality of the situation. One should use visuals to tell a true, complete and fast-paced story.

Data visualization is the last step in the data life cycle. This is where the data is processed for presentation in an easy-to-consume manner to the right audience for the right purpose. The data should be converted into a language and format that is best preferred and understood by the consumer of data. The presentation should aim to highlight the insights from the data in an actionable manner. If the data is presented in too much detail, then the consumer of that data might lose interest and the insight.

4.1 Excellence in Visualization

Data can be presented in the form of rectangular *tables*, or it can be presented in colorful graphs of various types. "Small, non-comparative, highly-labeled data sets usually belong in tables" – (Ed Tufte, 2001, p 33). However, as the amount of data grows, graphs are preferable. Graphics help give shape to data. Tufte, a pioneering expert on data visualization, presents the following objectives for graphical excellence:

1. *Show, and even reveal, the data*: The data should tell a story, especially a story hidden in large masses of data. However, reveal the data in context, so the story is correctly told.
2. *Induce the viewer to think of the substance of the data*: The format of the graph should be so natural to the data, that it hides itself and lets data shine.
3. *Avoid distorting what the data have to say*: Statistics can be used to lie. In the name of simplifying, some crucial context could be removed leading to distorted communication.
4. *Make large data sets coherent*: By giving shape to data, visualizations can help bring the data together to tell a comprehensivestory.
5. *Encourage the eyes to compare different pieces of data*: Organize the chart in ways the eyes would naturally move to derive insights from the graph.

6. *Reveal the data at several levels of detail*: Graphs leads to insights, which raise further curiosity, and thus presentations should help get to the root cause.
7. *Serve a reasonably clear purpose* – informing ordecision-making.
8. *Closely integrate with the statistical and verbal descriptions of the dataset*: There should be no separation of charts and text in presentation. Each mode should tell a

complete story. Intersperse text with the map/graphic to highlight the main insights.

Context is important in interpreting graphics. Perception of the chart is as important as the actual charts. Do not ignore the intelligence or the biases of the reader. Keep the template consistent, and only show variations in data. There can be many excuses for graphical distortion. E.g. "we are just approximating." Quality of information transmission comes prior to aesthetics of chart. Leaving out the contextual data can be misleading.

A lot of graphics are published because they serve a particular cause or a point of view. It is particularly important when in a for-profit or politically

contested environments. Many related dimensions can be folded into a graph. The more the dimensions that are represented in a graph, the richer and more useful the chart become. The data visualizer should understand the client's objects and present the data for accurate perception of the totality of the situation.

4.2 Types of Charts

There are many kinds of data as seen in the caselet above. Time series data is the most popular form of data. It helps reveal patterns over time. However, data could be organized around alphabetical list of things, such as countries or products or salespeople. Figure 5.2 shows some of the popular chart types and their usage.

1. *Line graph*. This is a basic and most popular type of displaying information. It shows data as a series of points connected by straight line segments. If mining with time-series data, time is usually shown on the x-axis. Multiple variables can be represented on the same scale ony-axis to compare of the line graphs of all the variables.
2. *Scatter plot:* This is another very basic and useful graphic form. It helps reveal the relationship between two variables. In the above caselet, it shows two dimensions: Life Expectancy and Fertility Rate. Unlike in a line graph, there are no line segments connecting thepoints.
3. *Bar graph:* A bar graph shows thin colorful rectangular bars with their lengths being proportional to the values represented. The bars can be plotted vertically or horizontally. The bar graphs use a lot of more ink than the line graph and should be

used when line graphs are inadequate.

4. *Stacked Bar graphs:* These are a particular method of doing bar graphs. Values of multiple variables are stacked one on top of the other to tell an interesting story. Bars can also be normalized such as the total height of every bar is equal, so it can show the relative composition of each bar.
5. *Histograms*: These are like bar graphs, except that they are useful in showing data frequencies or data values on classes (or ranges) of a numerical variable.

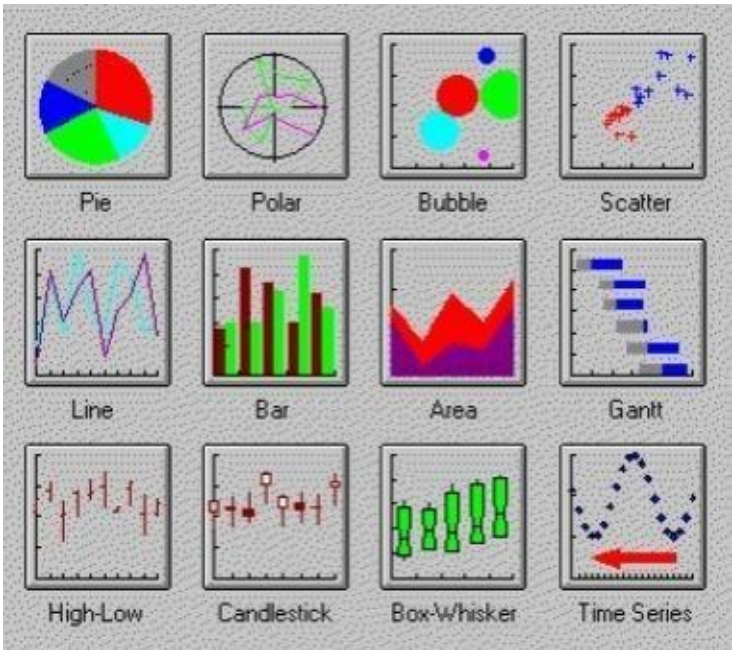


Figure 4.1: Many types of graphs

6. *Pie charts*: These are very popular to show the distribution of a variable, such as sales by region. The size of a slice is representative of the relative strengths of each value.
7. *Box charts:* These are special form of charts to show the distribution of variables. The box shows the middle half of the values, while whiskers on both sides extend to the extreme values in eitherdirection.
8. *Bubble Graph*: This is an interesting way of displaying multiple dimensions in one chart. It is a variant of a scatter plot with many data points marked on two dimensions. Now imagine that each data point on the graph is a bubble (or a circle) … the size of the circle and the color fill in the circle could represent two additional dimensions.

9. *Dials*: These are charts like the speed dial in the car, that shows whether the variable value (such as sales number) is in the low range, medium range, or high range. These ranges could be colored red, yellow and gree to give an instant view of the data.
10. *Geographical* Data maps are particularly useful maps to denote statistics. Figure 5.3 shows a tweet density map of the US. It shows where the tweets emerge from in the US.

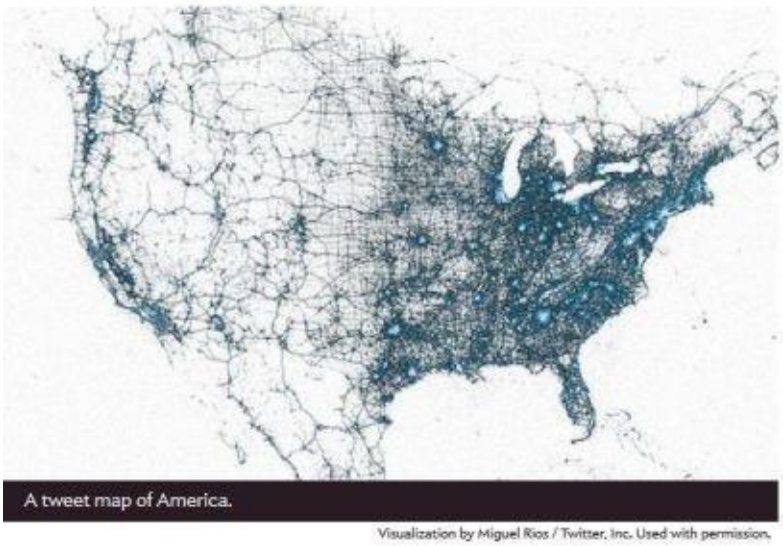


Figure 4.2: US tweet map (Source: Slate.com)

11. *Pictographs*: One can use pictures to represent data. E.g. Figure 4.2 shows the number of liters of water needed to produce one pound of each of the products, where images are used to show the product for easy reference. Each droplet of water also represents 50 liters ofwater.

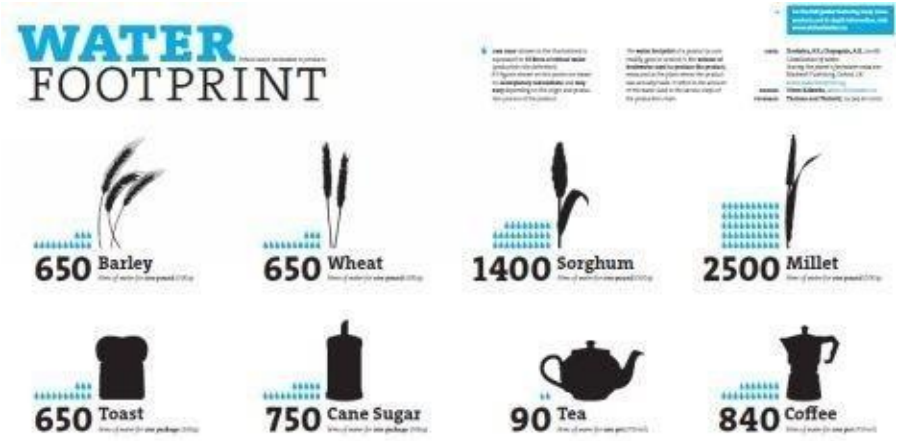


Figure 4.3: Pictograph of Water footprint (source : waterfootprint.org)

4.3 Tips for Data Visualization

To help the client in understanding the situation, the following considerations are important:

1. *Fetch appropriate and correct data for analysis*. This requires some understanding of the domain of the client and what is important for the client. E.g. in a business setting, one may need to understand the many measure of profitability and productivity.
2. *Sort the data in the most appropriate manner*. It could be sorted by numerical variables, or alphabetically by name.
3. *Choose appropriate method to present the data*. The data could be presented as a table, or it could be presented as any of the graph types.
4. *The data set could be pruned* to include only the more significant elements. More data is not necessarily better, unless it makes the most significant impact on the situation.
5. *The visualization could show additional dimension for reference* such as the expectations or targets with which to compare theresults.
6. *The numerical data may need to be binned into a few categories*. E.g. the orders per person were plotted as actual values, while the order sizes were binned into 4 categorical choices.
7. *High-level visualization could be backed by more detailed analysis.* For the most significant results, a drill-down may be required.
8. *There may be need to present additional textual information* to tell the whole story. For example, one may require notes to explain some extraordinary results.

Module 4

Chapter 1: Decision Trees

Decision trees are a simple way to guide one's path to a decision. The decision may be a simple binary one, whether to approve a loan or not. Or it may be a complex multi-valued decision, as to what may be the diagnosis for a particular sickness. Decision trees are hierarchically branched structures that help one come to a decision based on asking certain questions in a particular sequence. Decision trees are one of the most widely used techniques for classification. A good decision tree should be short and ask only a few meaningful questions. They are very efficient to use, easy to explain, and their classification accuracy is competitive with other methods. Decision trees can generate knowledge from a few test instances that can then be applied to a broad population. Decision trees are used mostly to answer relatively simple binary decisions.

1.1 Decision Tree problem

Imagine a conversation between a doctor and a patient. The doctor asks questions to determine the cause of the ailment. The doctor would continue to ask questions, till she is able to arrive at a reasonable decision. If nothing seems plausible, she might recommend some tests to generate more data and options.

This is how experts in any field solve problems. They use decision trees or decision rules. For every question they ask, the potential answers create separate branches for further questioning. For each branch, the expert would know how to proceed ahead. The process continues until the end of the tree is reached, which means a leaf node is reached.

Human experts learn from past experiences or data points. Similarly, a machine can be trained to learn from the past data points and extract some knowledge or rules from it. Decision trees use machine learning algorithmsto abstract knowledge from data. A decision tree would have a predictive accuracy based on how often it makes correct decisions.

1. The more training data is provided, the more accurate its knowledge extraction will be, and thus, it will make more accuratedecisions.
2. The more variables the tree can choose from, the greater is the likely of the accuracy of the decision tree.

3. In addition, a good decision tree should also be frugal so that it takes the least number of questions, and thus, the least amount of effort, to get to the right decision.

Here is an exercise to create a decision tree that helps make decisions about approving the play of an outdoor game. The objective is to predict the play decision given the atmospheric conditions out there. The decision is: Should the game be allowed or not? Here is the decision problem.

Outlook	**Temp**	**Humidity**	**Windy**	**Play**
Sunny	Hot	Normal	True	*??*

To answer that question, one should look at past experience, and see what decision was made in a similar instance, if such an instance exists. One could look up the database of past decisions to find the answer and try to come to an answer. Here is a list of the decisions taken in 14 instances of past soccer game situations. (Dataset courtesy: Witten, Frank, and Hall, 2010).

Outlook	**Temp**	**Humidity**	**Windy**	**Play**
Sunny	Hot	High	False	*No*
Sunny	Hot	High	True	*No*
Overcast	Hot	High	False	*Yes*
Rainy	Mild	High	False	*Yes*
Rainy	Cool	Normal	False	*Yes*
Rainy	Cool	Normal	True	*No*
Overcast	Cool	Normal	True	*Yes*

Sunny	Mild	High	False	*No*
Sunny	Cool	Normal	False	*Yes*
Rainy	Mild	Normal	False	*Yes*
Sunny	Mild	Normal	True	*Yes*
Overcast	Mild	High	True	*Yes*
Overcast	Hot	Normal	False	*Yes*
Rainy	Mild	High	True	*No*

If there were a row for Sunny/Hot/Normal/Windy condition in the data table, it would match the current problem; and the decision from that row could be

Used to answer the current problem. However, there is no such past instance in this case. There are three disadvantages of looking up the data table:

1. As mentioned earlier, how to decide if there isn't a row that corresponds to the exact situation today? If there is no exact matching instance available in the database, the past experience cannot guide the decision.
2. Searching through the entire past database may be time consuming, depending on the number of variables and the organization of the database.
3. What if the data values are not available for all the variables? In this instance, if the data for humidity variable was not available, looking up the past data would not help.

A better way of solving the problem may be to abstract the knowledge from the past data into decision tree or rules. These rules can be represented in a decision tree, and then that tree can be used make the decisions. The decision tree may not need values for all the variables.

1.2 Decision Tree Construction

A decision tree is a hierarchically branched structure. What should be the first question asked in creating the tree? One should ask the more important question first, and the less important

questions later. What is the most important question that should be asked to solve the problem? How is the importance of the questions determined? Thus, how should the root node of the tree be determined?

Determining root node of the tree: In this example, there are four choices based on the four variables. One could begin by asking one of the following questions: what is the outlook, what is the temperature, what is the humidity, and what is the wind speed? A criterion should be used to evaluate these choices. The key criterion would be that: which one of these questions gives the most insight about the situation? Another way to look at it would be the criterion of frugality. That is, which question will provide us the shortest ultimate decision tree? Another way to look at this is that if one is allowed to ask one and only one question, which one would one ask? In this case, the most important question should be the one that, by itself, helps make the most correct decisions with the fewest errors. The four questions can now be systematically compared, to see which variable by itself will help make the most correct decisions. One should systematically calculate the correctness of decisions based on each question. Then one can select the question with the most correct predictions, or the fewest errors.

Start with the first variable, in this case outlook. It can take three values, sunny, overcast, and rainy.

Start with the sunny value of outlook. There are five instances where the outlook is sunny. In 2 of the 5 instances the *play* decision was *yes*, and in the other three, the decision was No. Thus, if the decision rule was that Outlook:sunny → No, then 3 out of 5 decisions would be correct, while 2 out of 5 such decisions would be incorrect. There are 2 errors out of 5. This can be recorded in Row 1.

Attribute	**Rules**	**Error**	**Total Error**
Outlook	Sunny→No	2/5	

Similar analysis would be done for other values of the outlook variable. There are four instances where the outlook is overcast. In all 4 out 4 instances the Play decision was yes. Thus, if the decision rule was that Outlook:overcast→ Yes, then 4 out of 4 decisions would be

correct, while none of decisions would be incorrect. There are 0 errors out of 4. This can be recorded in the next row.

Attribute	**Rules**	**Error**	**Total Error**
Outlook	Sunny →No	2/5	
	Overcast →yes	0/4	

There are five instances where the outlook is rainy. In 3 of the 5 instances the *play* decision was *yes*, and in the other three, the decision was *no*. Thus, if the decision rule was that Outlook:rainy→ Yes, then 3 out of 5 decisions would be correct, while 2 out of 5 decisions would be incorrect. There will be 2/5 errors. This can be recorded in next row.

Attribute	**Rules**	**Error**	**Total Error**
Outlook	Sunny→No	2/5	4/14
	Overcast →yes	0/4	
	Rainy →yes	2/5	

Adding up errors for all values of outlook, there are 4 errors out of 14. In other words, Outlook gives 10 correct decisions out of 14, and 4 incorrect ones.

A similar analysis can be done for the other three variables. At the end of that analytical exercise, the following Error table will be constructed.

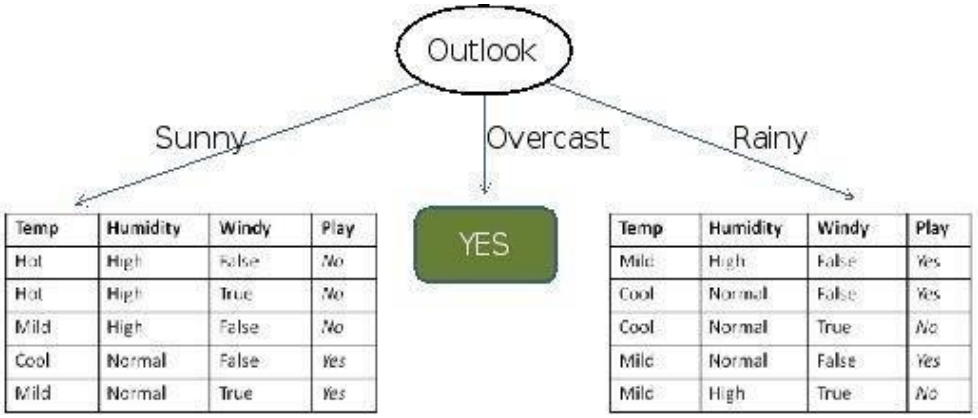
Attribute	**Rules**	**Error**	**Total Error**
Outlook	Sunny→No	2/5	4/14
	Overcast →yes	0/4	
	Rainy →yes	2/5	
Temp	Hot →No	2/4	5/14
	Mild →Yes	2/6	
	Cool → Yes	1/4	
Humidity	High → No	3/7	4/14
	Normal →Yes	1/7	
Windy	False →Yes	2/8	5/14
	True →No	3/6	

The variable that leads to the least number of errors (and thus the most number of correct decisions) should be chosen as the first node. In this case, two variables have the least number of errors. There is a tie between outlook and humidity, as both have 4 errors out of 14 instances. The tie can be broken using another criterion, the purity of resulting sub-trees.

If all the errors were concentrated in a few of the subtrees, and some of the branches were completely free of error, that is preferred from a usability perspective. Outlook has one error-free branch, for the overcast value, while there is no such pure sub-class for humidity variable. Thus the tie is broken in favor of outlook. The decision tree will use outlook as the first node, or the first splitting variable. The first question that should be asked to solve the Play problem, is 'What is the value of outlook'?

Splitting the Tree: From the root node, the decision tree will be split into three branches or sub-trees, one for each of the three values of outlook. Data for the root node (the entire data) will be divided into the three segments, one for each of the value of outlook. The sunny branch will inherit the data for the instances that had sunny as the value of outlook. These will be used for further building of that sub-tree. Similarly, the rainy branch will inherit data for the instances that had rainy as the value of outlook. These will be used for further building of that sub-tree. The overcast branch will inherit the data for the instances that had overcast as the outlook. However, there will be no need to build further on that branch. There is a clear decision, yes, for all instances when outlook value is overcast.

The decision tree will look like this after the first level of splitting.



Temp	Humidity	Windy	Play
Hot	High	False	No
Hot	High	True	No
Mild	High	False	No
Cool	Normal	False	Yes
Mild	Normal	True	Yes

Temp	Humidity	Windy	Play
Mild	High	False	Yes
Cool	Normal	False	Yes
Cool	Normal	True	No
Mild	Normal	False	Yes
Mild	High	True	No

Determining the next nodes of the tree: A similar recursive logic of tree building should be applied to each branch. For the sunny branch on the left, error values will be calculated for the three other variables – temp, humidity and windy. Final comparison looks like this:

Attribute	**Rules**	**Error**	**Total Error**
Temp	Hot->No	0/2	1/5
	Mild ->No	1/2	
	Cool -> yes	0/1	
Humidity	High->No	0/3	0/5
	Normal->Yes	0/2	
Windy	False->No	1/3	2/5
	True->Yes	1/2	

The variable of humidity shows the least amount of error, i.e. zero error. The other two variables have non-zero errors. Thus the Outlook:sunny branch on the left will use humidity as the next splitting variable.

Similar analysis should be done for the 'rainy' value of the tree. The analysis would look like this.

Attribute	**Rules**	**Error**	**Total Error**
Temp	Mild->Yes	1/3	2/5
	Cool->yes	1/2	

Humidity	High->No	1/2	2/5
	Normal->Yes	1/3	
Windy	False->Yes	0/3	0/5
	True-No	0/2	

For the Rainy branch, it can similarly be seen that the variable Windy gives all the correct answers, while none of the other two variables makes all the correct decisions This is how the final decision tree looks like. Here it is produced using Weka open-source data mining platform. This is the model that abstracts the knowledge of the past data of decision.

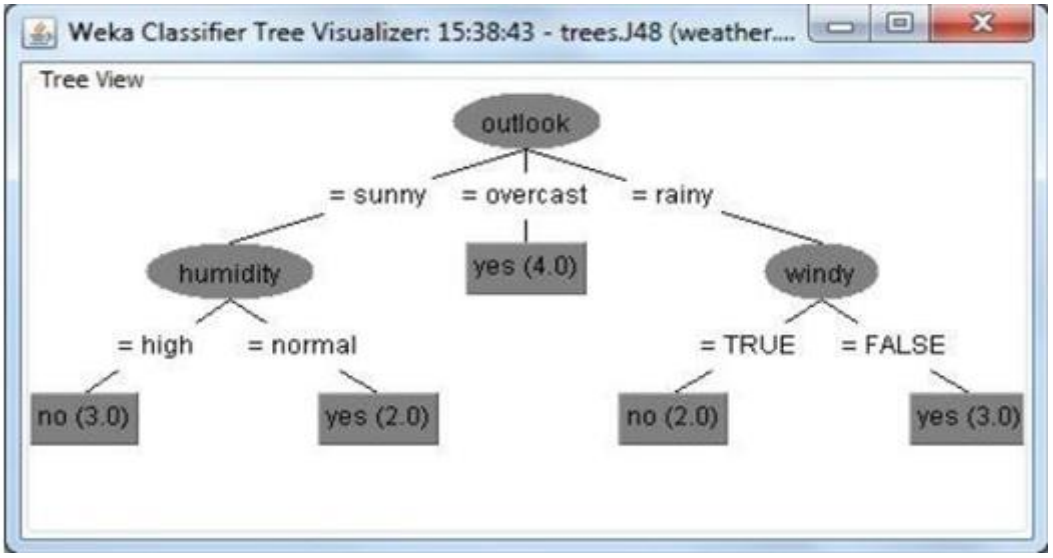


Figure 1.1: Decision Tree for the weather problem

This decision tree can be used to solve the current problem. Here is the problem again.

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	Normal	True	??

According to the tree, the first question to ask is about outlook. In this problem the outlook is sunny. So, the decision problem moves to the Sunny branch of the tree. The node in that sub-tree is humidity. In the problem, Humidity is Normal. That branch leads to an answer Yes. Thus, the answer to the play problem is Yes.

Outlook	**Temp**	**Humidity**	**Windy**	**Play**
Sunny	Hot	Normal	True	***Yes***

Lessons from constructing trees

Here are some benefits of using this decision tree compared with looking up the answers from the data table (Figure 6.1)

	Decision Tree	**Table Lookup**
Accuracy	Varied level of accuracy	100% accurate
Generality	General. Applies to all situations	Applies only when a similar case had occurred earlier
Frugality	Only three variables needed	All four variables are needed
Simple	Only one, or max two variable values are needed	All four variable values are needed
Easy	Logical, and easy to understand	Can be cumbersome to look up; no understanding of the logic behind the decision

Figure 1.1: Comparing Decision Tree with Table Look-up

Here are a few observations about how the tree was constructed:

1. The final decision tree has zero errors in mapping to the prior data. In other words, the tree has a *predictive accuracy of 100%*. The tree completely fits the data. In real life situations, such perfect predictive accuracy is not possible when making decision trees. When there are larger, complicated data sets, with many more variables, a perfect fit is unachievable. This is especially true in business and social contexts, where things are not always fully clear and consistent.
2. The decision tree algorithm *selected the minimum number of variables* that are needed to solve the problem. Thus, one can start with all available data variables, and let the decision-tree algorithm select the ones that are useful, and discard the rest.
3. This tree is *almost symmetric* with all branches being of almost similar lengths. However, in real life situations, some of the branches may be much longer than the others, and the tree may need to be pruned to make it more balanced and usable.
4. It may be possible to *increase predictive accuracy by making more sub- trees* and making the tree longer. However, the marginal accuracy gained from each subsequent level in the tree will be less, and may not be worth the loss in ease and interpretability of the tree. If the branches are long and complicated, it will be difficult to understand and use. The longer branches may need to be trimmed to keep the tree easy touse.
5. A perfectly fitting tree has the *danger of over-fitting the data*, thus capturing all the random variations in the data. It may fit the training data well, but may not do well in predicting the future real instances.
6. There was a *single best tree* for this data. There could however be two or more equally efficient decision trees of similar length with similar predictive accuracy for the same data set. Decision trees are *based strictly on patterns within the data*, and do not rely on any underlying theory of the problem domain. When multiple candidate trees are available, one could choose whichever is easier to understand, communicate or implement.

1.3 Decision Tree Algorithms

As we saw, decision trees employ the divide and conquer method. The data is branched at each node according to certain criteria until all the data is assigned to leaf nodes. It recursively divides a training set until each division consists of examples from one class.

The following is a pseudo code for making decision trees:

1. Create a root node and assign all of the training data to it.
2. Select the best splitting attribute according to certain criteria.
3. Add a branch to the root node for each value of the split.
4. Split the data into mutually exclusive subsets along the lines of the specific split.
5. Repeat steps 2 and 3 for each and every leaf node until a stopping criteria is reached.

There are many algorithms for making decision trees. Decision tree algorithms differ on three key elements:

1. **Splitting criteria**
    1. Which variable to use for the first split? How should one determine the most important variable for the first branch, and subsequently, for each sub-tree? There are many measures like least errors, information gain, gini's coefficient, etc.
    2. What values to use for the split? If the variables have continuous values such as for age or blood pressure, what value-ranges should be used to make bins?
    3. How many branches should be allowed for each node? There could be binary trees, with just two branches at each node. Or there could be more branches allowed.
2. **Stopping criteria:** When to stop building the tree? There are two major ways to make that determination. The tree building could be stopped when a certain depth of the branches has been reached and the tree becomes unreadable after that. The tree could also be stopped when the error level at any node is within predefined tolerable levels.
3. **Pruning:** The tree could be trimmed to make it more balanced and more easily usable. The pruning is often done after the tree is constructed, to balance out the tree and improve usability. The symptoms of an over- fitted tree are a tree too deep, with too many branches, some of which may reflect anomalies due to noise or outliers. Thus, the tree should be pruned. There are two approaches to avoid over-fitting.

Pre-pruning means to halt the tree construction early, when certain criteria are met. The downside is that it is difficult to decide what criteria to use for halting the construction, because we do not know what may happen subsequently, if we keep growing the tree.

Post-pruning: Remove branches or sub-trees from a "fully grown" tree. This method is commonly used. C4.5 algorithm uses a statistical method to estimate the errors at each node for pruning. A validation set may be used for pruning as well.

Comparison of different Decision Tree Algorithms

Decision-Tree	C4.5	CART	CHAID
Full Name	Iterative Dichotomiser (ID3)	Classification and Regression Trees	Chi-square Automatic Interaction Detector
Basic algorithm	Hunt's algorithm	Hunt's algorithm	adjusted significance testing
Developer	Ross Quinlan	Bremman	Gordon Kass
When developed	1986	1984	1980
Types of trees	Classification	Classification & Regression trees	Classification & regression
Serial implementation	Tree-growth & Tree-pruning	Tree-growth & Tree-pruning	Tree-growth & Tree-pruning
Type of data	Discrete & Continuous; Incomplete data	Discrete and Continuous	Non-normal data also accepted
Types of splits	Multi-way splits	Binary splits only; Clever surrogate splits to reduce tree depth	Multi-way splits as default
Splitting criteria	Information gain	Gini's coefficient, and others	*Chi*-square test
Pruning Criteria	Clever bottom-up technique avoids overfitting	Remove weakest links first	Trees can become very large
Implementation	Publicly available	Publicly available in most packages	Popular in market research, for segmentation

Chapter 2: Regression

Regression is a well-known statistical technique to model the predictive relationship between several independent variables (DVs) and one dependent variable. The objective is to find the best-fitting curve for a dependent variable in a multidimensional space, with each independent variable being a dimension. The curve could be a straight line, or it could be a nonlinear curve. The quality of fit of the curve to the data can be measured by a coefficient of correlation (r), which is the square root of the amount of variance explained by the curve.
The key steps for regression are simple:

1. List all the variables available for making the model.
2. Establish a Dependent Variable (DV) of interest.
3. Examine visual (if possible) relationships between variables of interest.
4. Find a way to predict DV using the other variables.

2.1 Correlations and Relationships

Statistical relationships are about which elements of data hang together, and which ones hang separately. It is about categorizing variables that have a relationship with one another, and categorizing variables that are distinct and unrelated to other variables. It is about describing significant positive relationships and significant negative differences.

The first and foremost measure of the strength of a relationship is co-relation (or correlation). The strength of a correlation is a quantitative measure that is measured in a normalized range between 0 (zero) and 1. A correlation of 1 indicates a perfect relationship, where the two variables are in perfect sync. A correlation of 0 indicates that there is no relationship between the variables.

The relationship can be positive, or it can be an inverse relationship, that is, the variables may move together in the same direction or in the opposite direction. Therefore, a good measure of correlation is the correlation coefficient, which is the square root of correlation. This coefficient, called r, can thus range from −1 to +1. An r value of 0 signifies no relationship. An r value of 1 shows perfect relationship in the same direction, and an r value of
−1 shows a perfect relationship but moving in opposite directions.

$$r = \frac{[(x - \bar{x})][(y - \bar{y})]}{\sqrt{[(x - \bar{x})^2][(y - \bar{y})^2]}}$$

Given two numeric variables x and y, the coefficient of correlation r is mathematically computed by the following equation. $\bar{x}$ (called x-bar) is the mean of x, and $\bar{y}$ (y-bar) is the mean of y.

2.2 Visual look at relationships

A scatter plot (or scatter diagram) is a simple exercise for plotting all data points between two variables on a two-dimensional graph. It provides a visual layout of where all the data points are placed in that two-dimensional space. The scatter plot can be useful for graphically intuiting the relationship between two variables.

Here is a picture (Figure 2.1) that shows many possible patterns in scatter diagrams.

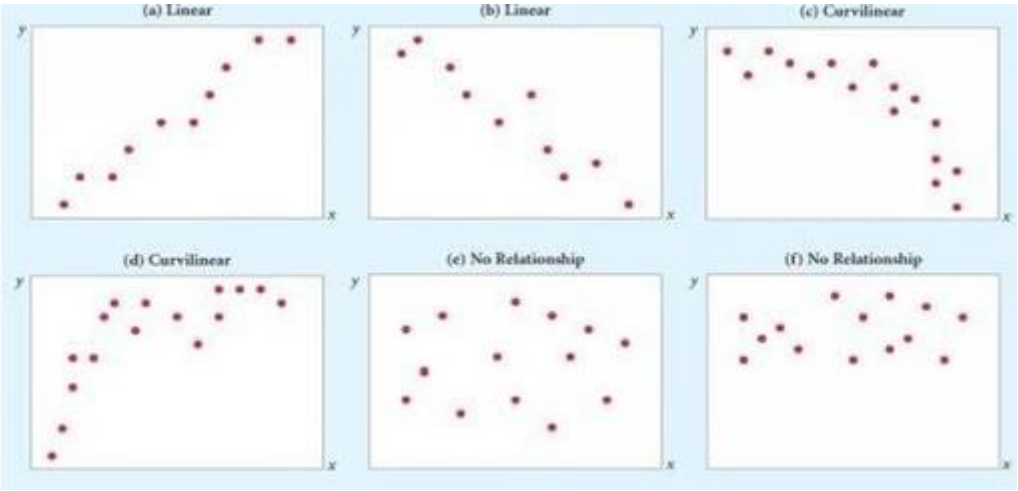


Figure 2.1: Scatter plots showing types of relationships among two variables

Chart (a) shows a very strong linear relationship between the variables *x* and *y*. That means the value of *y* increases proportionally with *x*. Chart (b) also shows a strong linear relationship between the variables *x* and *y*. Here it is an inverse relationship. That means the value of *y* decreases proportionally with *x*.

Chart (c) shows a curvilinear relationship. It is an inverse relationship, which means that the value of *y* decreases proportionally with *x*. However, it seems a relatively well-defined relationship, like an arc of a circle, which can be represented by a simple quadratic equation (quadratic means the power of two, that is, using terms like x^2 and y^2). Chart (d) shows a

positive curvilinear relationship. However, it does not seem to resemble a regular shape, and thus would not be a strong relationship. Charts (e) and (f) show no relationship. That means variables *x* and *y* are independent of each other.

Charts (a) and (b) are good candidates that model a simple linear regression model (the terms regression model and regression equation can be used

interchangeably). Chart (c) too could be modeled with a little more complex, quadratic regression equation. Chart (d) might require an even higher order polynomial regression equation to represent the data. Charts (e) and (f) have no relationship, thus, they cannot be modeled together, by regression or using any other modeling tool.

2.3 Regression Exercise

The regression model is described as a linear equation that follows. *y* is the dependent variable, that is, the variable being predicted. *x* is the independent variable, or the predictor variable. There could be many predictor variables (such as $x_1, x_2, \ldots$) in a regression equation. However, there can be only one dependent variable (*y*) in the regression equation.

$y = \beta_0 + \beta_1 x + \varepsilon$

A simple example of a regression equation would be to predict a house price from the size of the house. Here is a sample house prices data:

House Price	Size (sqft)
$229,500	1850
$273,300	2190
$247,000	2100
$195,100	1930

$261,000	2300
$179,700	1710
$168,500	1550
$234,400	1920
$168,800	1840
$180,400	1720
$156,200	1660
$288,350	2405
$186,750	1525
$202,100	2030
$256,800	2240

The two dimensions of (one predictor, one outcome variable) data can be plotted on a scatter diagram. A scatter plot with a best-fitting line looks like the graph that follows (Figure 6.2).

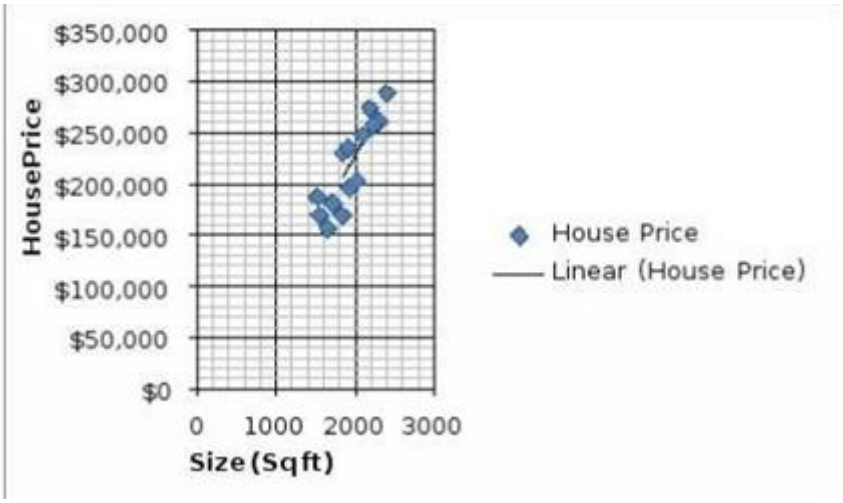


Figure 2.2: Scatter plot and regression equation between House price and house size.

Visually, one can see a positive correlation between House Price and Size (sqft). However, the relationship is not perfect. Running a regression model between the two variables produces the following output (truncated).

Regression Statistics	
r	**0.891**
r^2	**0.794**
	Coefficients
Intercept	**-54191**
Size (sqft)	**139.48**

It shows the coefficient of correlation is 0.891. r^2, the measure of total variance explained by the equation, is 0.794, or 79%. That means the two variables are moderately and positively correlated. Regression coefficients help create the following equation for predicting house prices.

House Price ($) = 139.48 * Size(sqft) – 54191

This equation explains only 79% of the variance in house prices. Suppose other predictor variables are made available, such as the number of rooms in the house. It might help improve the regression model.

The house data now looks like this:

House Price	Size (sqft)	#Rooms
$229,500	1850	4
$273,300	2190	5
$247,000	2100	4
$195,100	1930	3
$261,000	2300	4
$179,700	1710	2

$168,500	1550	2
$234,400	1920	4
$168,800	1840	2
$180,400	1720	2
$156,200	1660	2
$288,350	2405	5
$186,750	1525	3
$202,100	2030	2
$256,800	2240	4

While it is possible to make a 3-dimensional scatter plot, one can alternatively examine the correlation matrix among the variables.

	House Price	*Size (sqft)*	*#Rooms*
House Price	1		
Size (sqft)	0.891	1	
Rooms	0.944	0.748	1

It shows that the House price has a strong correlation with number of rooms (0.944) as well. Thus, it is likely that adding this variable to the regression model will add to the strength of the model.

Running a regression model between these three variables produces the following output (truncated).

Regression Statistics	
r	**0.984**
r^2	**0.968**
	Coefficients
Intercept	**12923**
Size(sqft)	**65.60**
Rooms	**23613**

It shows the co-efficient of correlation of this regression model is 0.984. R^2, the total variance explained by the equation, is 0.968 or 97%. That means the variables are positively and very strongly correlated. Adding a new relevant variable has helped improve the strength of the regression model.

Using the regression coefficients helps create the following equation for predicting house prices.

House Price ($) = 65.6 * Size (sqft) + 23613 * Rooms + 12924

This equation shows a 97% goodness of fit with the data, which is very good for business and economic data. There is always some random variation in naturally occurring business data, and it is not desirable to overfit the model to the data.

This predictive equation should be used for future transactions. Given a situation as below, it will be possible to predict the price of the house with 2000 sq ft and 3 rooms.

House Price	Size (sqft)	#Rooms
??	2000	3

House Price ($) = 65.6 * 2000 (sqft) + 23613 * 3 + 12924 = $214,963

The predicted values should be compared with the actual values to see how close the model is able to predict the actual value. As new data points become available, there are opportunities to fine-tune and improve themodel.

2.4 Non-linear regression exercise

The relationship between the variables may also be curvilinear. For example, given past data from electricity consumption (KwH) and temperature (temp), the objective is to predict the electrical consumption from the temperature value. Here are a dozen past observations.

KWatts	Temp (F)
12530	46.8
10800	52.1
10180	55.1
9730	59.2
9750	61.9
10230	66.2
11160	69.9
13910	76.8
15690	79.3
15110	79.7

17020	80.2
17880	83.3

In two dimensions (one predictor, one outcome variable) data can be plotted on a scatter diagram. A scatter plot with a best-fitting line looks like the graph below (Figure 7.3).

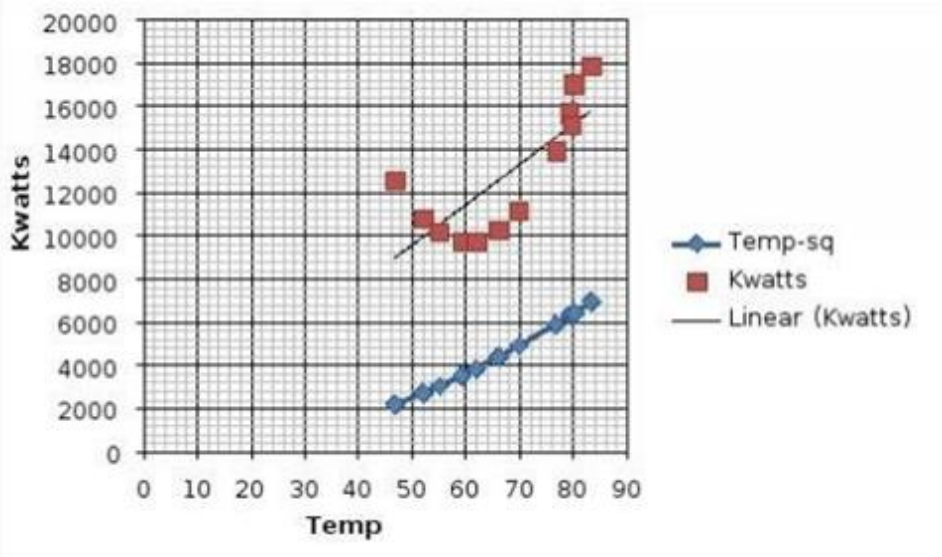


Figure 2.2: Scatter plots showing regression between (a) kwatts and temp, and (b) kwatts and temp square

It is visually clear that the first line does not fit the data well. The relationship between temperature and Kwatts follows a curvilinear model, where it hits bottom at a certain value of temperature. The regression model confirms the relationship since R is only 0.77 and R- square is also only 60%. Thus, only 60% of the variance is explained.

The regression model can then be enhanced using a $Temp^2$ variable in the equation. The second line is the relationship between KWH and $Temp^2$. The scatter plot shows that the Energy consumption shows a strong linear relationship with the quadratic $Temp^2$ variable. Running the regression model after adding the quadratic variable, leads to the following results:

Regression Statistics	
r	0.992
r^2	0.984
	Coefficients
Intercept	**67245**
Temp (F)	**-1911**
Temp-sq	**15.87**

It shows that the co-efficient of correlation of the regression model is now

0.99. R^2, the total variance explained by the equation is 0.985, or 98.5%. That means the variables are very strongly and positively correlated. The regression coefficients help create the following equation for

Energy Consumption (Kwatts) = 15.87 * Temp^2 -1911 * Temp + 67245

This equation shows a 98.5% fit which is very good for business and economic contexts. Now one can predict the Kwatts value for when the temperature is 72-degrees.

Energy consumption = (15.87 * 72*72) - (1911 * 72) + 67245 = 11923 Kwatts

2.5 Logistic Regression

Regression models traditionally work with continuous numeric value data for dependent and independent variables. Logistic regression models can, however, work with dependent variables with binary values, such as whether a loan is approved (yes or no). Logistic regression measures the relationship between a categorical dependent variable and one or more independent variables. For example, Logistic regression might be used to predict whether a patient has a given disease (e.g. diabetes), based on observed characteristics of the patient (age, gender, body mass index, results of blood tests, etc.).

Logistical regression models use probability scores as the predicted values of the dependent variable. Logistic regression takes the natural logarithm of the odds of the dependent variable

being a case (referred to as the logit) to create a continuous criterion as a transformed version of the dependent variable. Thus the logit transformation is used in logistic regression as the dependent variable. The net effect is that although the dependent variable in logistic regression is binomial (or categorical, i.e. has only two possible values), the logit is the continuous function upon which linear regression is conducted. Here is the general logistic function, with independent variable on the horizontal axis and the logit dependent variable on the vertical axis (Figure 2.3).

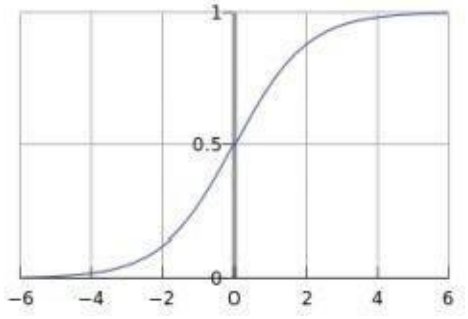


Figure 2.3: General Logit function

All popular data mining platforms provide support for regular multiple regression models, as well as options for Logistic Regression.

Advantages and Disadvantages of Regression Models

Regression Models are very popular because they offer many advantages.

1. Regression models are easy to understand as they are built upon basic statistical principles such as correlation and least square error.
2. Regression models provide simple algebraic equations that are easy to understand and use.
3. The strength (or the goodness of fit) of the regression model is measured in terms of the correlation coefficients, and other related statistical parameters that are well understood.
4. Regression models can match and beat the predictive power of other modeling techniques.
5. Regression models can include all the variables that one wants to include in the model.
6. Regression modeling tools are pervasive. They are found in statistical packages as well as data mining packages. MS Excel spreadsheets can also provide simple regression modeling capabilities.

Regression models can however prove inadequate under many circumstances.

1. Regression models can not cover for poor data quality issues. If the data is not prepared well to remove missing values, or is not well-behaved in terms of a normal distribution, the validity of the modelsuffers.
2. Regression models suffer from collinearity problems (meaning strong linear correlations among some independent variables). If the independent variables have strong correlations among themselves, then they will eat into each other's predictive power and the regression coefficients will lose their ruggedness. Regression models will not automatically choose between highly collinear variables, although some packages attempt to do that.
3. Regression models can be unwieldy and unreliable if a large number of variables are included in the model. All variables entered into the model will be reflected in the regression equation, irrespective of their contribution to the predictive power of the model. There is no concept of automatic pruning of the regression model.
4. Regression models do not automatically take care of non-linearity. The user needs to imagine the kind of additional terms that might be needed to be added to the regression model to improve its fit.
5. Regression models work only with numeric data and not with categorical variables. There are ways to deal with categorical variables though by

   Creating multiple new variables with a yes/no value.

Chapter 3: Artificial Neural Networks

Artificial Neural Networks (ANN) are inspired by the information processing model of the mind/brain. The human brain consists of billions of neurons that link with one another in an intricate pattern. Every neuron receives information from many other neurons, processes it, gets excited or not, and passes its state information to other neurons.

Just like the brain is a multipurpose system, so also the ANNs are very versatile systems. They can be used for many kinds of pattern recognition and prediction. They are also used for classification, regression, clustering, association, and optimization activities. They are used in finance, marketing, manufacturing, operations, information systems applications, and soon.

ANNs are composed of a large number of highly interconnected processing elements (neurons) working in a multi-layered structures that receive inputs, process the inputs, and produce an output. An ANN is designed for a specific application, such as pattern recognition or data classification, and trained through a learning process. Just like in biological systems, ANNs make adjustments to the synaptic connections with each learning instance.

ANNs are like a black box trained into solving a particular type of problem, and they can develop high predictive powers. Their intermediate synaptic parameter values evolve as the system obtains feedback on its predictions, and thus an ANN learns from more training data (Figure 8.1).

Figure 3.1 General ANN Model

3.1 Business Applications of ANN

Neural networks are used most often when the objective function is complex, and where there exists plenty of data, and the model is expected to improve over a period of time. A few sample applications:

1. They are used in stock price prediction where the rules of the game are extremely

complicated, and a lot of data needs to be processed very quickly.

2. They are used for character recognition, as in recognizing hand-written text, or damaged or mangled text. They are used in recognizing finger prints. These are complicated patterns and are unique for each person. Layers of neurons can progressively clarify the pattern leading to a remarkably accurate result.
3. They are also used in traditional classification problems, like approving a financial loan application.

3.2 Design Principles of an Artificial Neural Network

1. A neuron is the basic processing unit of the network. The neuron (or processing element) receives inputs from its preceding neurons (or PEs), does some nonlinear weighted computation on the basis of those inputs, transforms the result into its output value, and then passes on the output to the next neuron in the network (Figure 8.2). X's are the inputs, w's are the weights for each input, and y is the output.

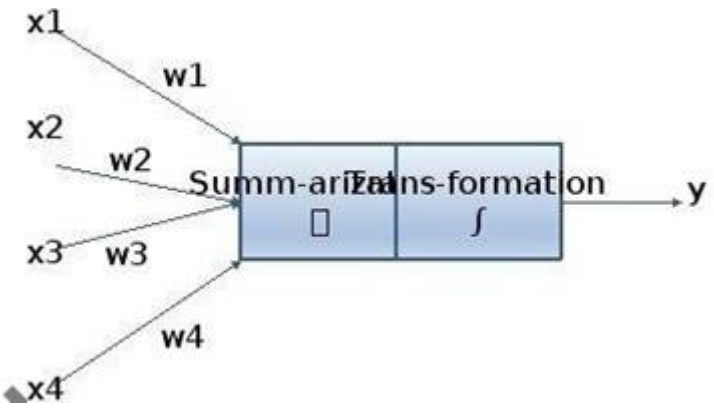


Figure: shows the model

2. A Neural network is a multi-layered model. There is at least one input neuron, one output neuron, and at least one processing neuron. An ANN with just this basic structure would be a simple, single-stage computational unit. A simple task may be processed by just that one neuron and the result may be communicated soon. ANNs however, may have multiple layers of processing elements in sequence. There could be many neurons involved in a sequence depending upon the complexity of the predictive action. The layers of PEs could work in sequence, or they could work in parallel.

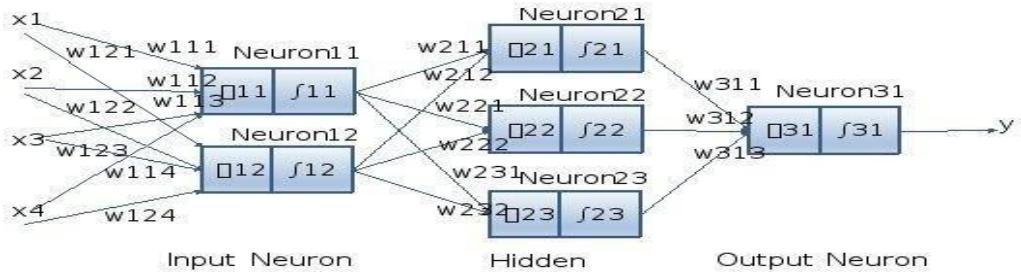


Figure 3.3: Model for a multi-layer ANN

3. The processing logic of each neuron may assign different weights to the various incoming input streams. The processing logic may also use non- linear transformation, such as a sigmoid function, from the processed values to the output value. This processing logic and the intermediate weight and processing functions are just what works for the system as a whole, in its objective of solving a problem collectively. Thus, neural networks are considered to be an opaque and a black-boxsystem.

4. The neural network can be trained by making similar decisions over and over again with many training cases. It will continue to learn by adjusting its internal computation and communication based on feedback about its previous decisions. Thus, the neural networks become better at making a decision as they handle more and more decisions.

Depending upon the nature of the problem and the availability of good training data, at some point the neural network will learn enough and begin to match the predictive accuracy of a human expert. In many practical situations, the predictions of ANN, trained over a long period of time with a large number of training data, have begun to decisively become more accurate than human experts. At that point ANN can begin to be seriously considered for deployment in real situations in real time.

3.3 Representation of a Neural Network

A neural network is a series of neurons that receive inputs from other neurons. They do a weighted summation function of all the inputs, using different weights (or importance) for each input. The weighted sum is then transformed into an output value using a transfer function.

Learning in ANN occurs when the various processing elements in the neural network adjust the underlying relationship (weights, transfer function, etc) between input and outputs, in

response to the feedback on their predictions. If the prediction made was correct, then the weights would remain the same, but if the prediction was incorrect, then the parameter values would change.

The Transformation (Transfer) Function is any function suitable for the task at hand. The transfer function for ANNs is usually a non-linear sigmoid function. Thus, if the normalized computed value is less than some value (say 0.5) then the output value will be zero. If the computed value is at the cut-off threshold, then the output value will be a 1. It could be a nonlinear hyperbolic function in which the output is either a -1 or a 1. Many other functions could be designed for any or all of the processing elements.

Thus, in a neural network, every processing element can potentially have a different number of input values, a different set of weights for those inputs, and a different transformation function. Those values support and compensate for one another until the neural network as a whole learns to provide the correct output, as desired by the user.

3.4 Architecting a Neural Network

There are many ways to architect the functioning of an ANN using fairly simple and open rules with a tremendous amount of flexibility at each stage. The most popular architecture is a Feed-forward, multi-layered perceptron with back-propagation learning algorithm. That means there are multiple layers of PEs in the system and the output of neurons are fed forward to the PEs in the next layers; and the feedback on the prediction is fed back into the neural network for learning to occur. This is essentially what was described in the earlier paragraphs. ANN architectures for different applications are shown in Table 8.1.

Classification	Feedforward networks (MLP), radial basis function, and probabilistic
Regression	Feedforward networks (MLP), radial basis function
Clustering	Adaptive resonance theory (ART), Self-organizing maps (SOMs)
Association Rule Mining	Hopfield networks

Table 3.1: ANN architectures for different applications

3.5 Steps required building an ANN

1. Gather data and Divide into training data and test data. The training data needs to be further divided into training data and validationdata.
2. Select the network architecture, such as Feed forward network.
3. Select the algorithm, such as Multi-layer Perception.
4. Set network parameters.
5. Train the ANN with training data.
6. Validate the model with validation data.
7. Freeze the weights and other parameters.
8. Test the trained network with test data.
9. Deploy the ANN when it achieves good predictive accuracy.

Training an ANN requires that the training data be split into three parts (Table 3.2):

Training set	This data set is used to adjust the weights on the neural network (~ 60%).
Validation set	This data set is used to minimize overfitting and verifying accuracy (~ 20%).
Testing set	This data set is used only for testing the final solution in order to confirm the actual predictive power ofthe network (~ 20%).
k-fold cross-validation	This approach means that the data is divided into k equal pieces, and the learning process is repeated k- times with each pieces becoming the training set. This process leads to less bias and more accuracy, butis more time consuming.

Table 3.2: ANN Training datasets

3.6 Advantages and Disadvantages of using ANNs

There are many benefits of using ANN.

1. ANNs impose very little restrictions on their use. ANN can deal with (identify/model) highly nonlinear relationships on their own, without much work from the user or analyst. They help find practical data-driven solutions where algorithmic solutions are non-existent or too complicated.
2. There is no need to program neural networks, as they learn from examples. They get better with use, without much programming effort.
3. They can handle a variety of problem types, including classification, clustering, associations, etc.
4. ANN are tolerant of data quality issues and they do not restrict the data to follow strict normality and/or independence assumptions.
5. They can handle both numerical and categorical variables.
6. ANNs can be much faster than other techniques.
7. Most importantly, they usually provide better results (prediction and/or clustering) compared to statistical counterparts, once they have been trained enough.

The key disadvantages arise from the fact that they are not easy to interpret or explain or compute.

1. They are deemed to be black-box solutions, lacking explainability. Thus they are difficult to communicate about, except through the strength of their results.
2. Optimal design of ANN is still an art: it requires expertise and extensive experimentation.
3. It can be difficult to handle a large number of variables (especially the rich nominal attributes).
4. It takes large data sets to train an ANN.

Chapter 4: Cluster Analysis

Cluster analysis is used for automatic identification of natural groupings of things. It is also known as the segmentation technique. In this technique, data instances that are similar to (or near) each other are categorized into one cluster. Similarly, data instances that are very different (or far away) from each other are moved into different clusters.

Clustering is an unsupervised learning technique as there is no output or dependent variable for which a right or wrong answer can be computed. The correct number of clusters or the definition of those clusters is not known ahead of time. Clustering techniques can only suggest to the user how many clusters would make sense from the characteristics of the data. The user can specify a different, larger or smaller, number of desired clusters based on their making business sense. The cluster analysis technique will then define many distinct clusters from analysis of the data, with cluster definitions for each of those clusters. However, there are good cluster definitions, depending on how closely the cluster parameters fit the data.

4.1 Applications of Cluster Analysis

Cluster analysis is used in almost every field where there is a large variety of transactions. It helps provide characterization, definition, and labels for populations. It can help identify natural groupings of customers, products, patients, and so on. It can also help identify outliers in a specific domain and thus decrease the size and complexity of problems. A prominent business application of cluster analysis is in market research. Customers are segmented into clusters based on their characteristics—wants and needs, geography, price sensitivity, and so on. Here are some examples of clustering:

1. *Market Segmentation*: Categorizing customers according to their similarities, for instance by their common wants and needs, and propensity to pay, can help with targeted marketing.
2. *Product portfolio*: People of similar sizes can be grouped together to make small, medium and large sizes for clothing items.
3. *Text Mining*: Clustering can help organize a given collection of text documents according to their content similarities into clusters of related topics.

4.2 Definition of a Cluster

An operational definition of a cluster is that, given a representation of n objects, find K groups based on a measure of similarity, such that objects within the same group are alike but the objects in different groups are not alike.

However, the notion of similarity can be interpreted in many ways. Clusters can differ in terms of their shape, size, and density. Clusters are patterns, and there can be many kinds of patterns. Some clusters are the traditional types, such as data points hanging together. However, there are other clusters, such as all points representing the circumference of a circle. There may be concentric circles with points of different circles representing different clusters. The presence of noise in the data makes the detection of the clusters even more difficult.

An ideal cluster can be defined as a set of points that is compact and isolated. In reality, a cluster is a subjective entity whose significance and interpretation requires domain knowledge. In the sample data below (Figure 9.1), how many clusters can one visualize?

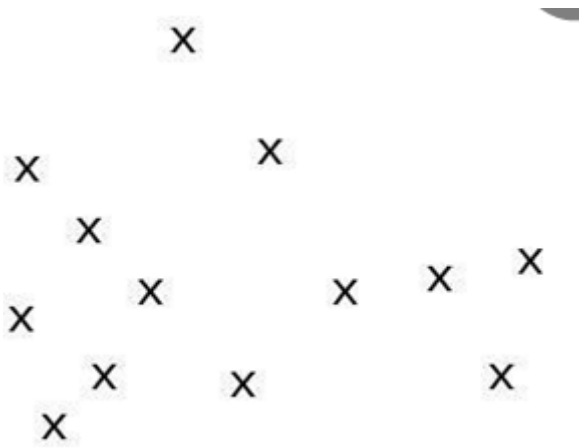


Figure 4.1: Visual cluster example

It seems like there are two clusters of approximately equal sizes. However, they can be seen as three clusters, depending on how we draw the dividing lines. There is not a truly optimal way to calculate it. Heuristics are often used to define the number of clusters.

4.4 Representing clusters

The clusters can be represented by a central or modal value. A cluster can be defined as the *centroid* of the collection of points belonging to it. A *centroid* is a measure of central tendency. It is the point from where the sum total of squared distance from all the points is the minimum. A real-life equivalent would be the city center as the point that is considered the most easy to use by all constituents of the city. Thus all cities are defined by their centers or downtown areas.

A cluster can also be represented by the most frequently occurring value in the cluster, i.e. the cluster can defined by its modal value. Thus, a particular cluster representing a social point of view could be called the 'soccer moms', even though not all members of that cluster need currently be a mom with soccer-playing children.

4.5 Clustering techniques

Cluster analysis is a machine-learning technique. The quality of a clustering result depends on the *algorithm*, the *distance* function, and the *application*. First, consider the distance function. Most cluster analysis methods use a distance measure to calculate the closeness between pairs of items. There are two major measures of distances: Euclidian distance ("as the crow flies" or straight line) is the most intuitive measure. The other popular measure is the Manhattan (rectilinear) distance, where one can go only in orthogonal directions. The Euclidian distance is the hypotenuse of a right triangle, while the Manhattan distance is the sum of the two legs of the righttriangle.

In either case, the key objective of the clustering algorithm is the same:

Inter-clusters distanceÞ maximized; and

Intra-clusters distanceÞ minimized

There are many algorithms to produce clusters. There are top-down, hierarchical methods that start with creating a given number of best-fitting clusters. There are also bottom-up methods that begin with identifying naturally occurring clusters.

The most popular clustering algorithm is the K-means algorithm. It is a top- down, statistical technique, based on the method of minimizing the least squared distance from the center points of the clusters. Other techniques, such as neural networks, are also used for clustering.

Comparing cluster algorithms is a difficult task as there is no single right number of clusters. However, the speed of the algorithm and its versatility in terms of different dataset are important criteria.

Here is the generic pseudocode for clustering

1. *Pick an arbitrary number of groups/segments to be created*
2. *Start with some initial randomly-chosen center values forgroups*
3. *Classify instances to closest groups*
4. *Compute new values for the group centers*
5. *Repeat step 3 & 4 till groups converge*
6. *If clusters are not satisfactory, go to step 1 and pick a different number of groups/segments*

The clustering exercise can be continued with a different number of clusters and different location of those points. Clusters are considered good if the cluster definitions stabilize, and the stabilized definitions prove useful for the purpose at hand. Else, repeat the clustering exercise with a different number of clusters, and different starting points for group means.

4.6 K-Means Algorithm for clustering

K-means is the most popular clustering algorithm. It iteratively computes the clusters and their centroids. It is a top down approach to clustering. Starting with a given number of K clusters, say 3 clusters. Thus three random centroids will be created as starting points of the centers of three clusters. The circles are initial cluster centroids

Step 1: For a data point, distance values will be from each of the three centroids. The data point will be assigned to the cluster with the shortest distance to the centroid. All data points will thus, be assigned to one data point or the other .The arrows from each data element shows the centroid that the point is assigned to.

Step 2: The centroid for each cluster will now be recalculated such that it is closest to all the data points allocated to that cluster. The dashed arrows show the centroids being moved from their old (shaded) values to the revised new values (Figure 9.7).

Step 3: Once again, data points are assigned to the three centroids closest to it (Figure 9.8).

The new centroids will be computed from the data points in the cluster until finally, the centroids stabilize in their locations. These are the three clusters computed by this algorithm.

These cluster definitions are different from the ones derived visually. This is a function of the random starting centroid values. The centroid points used earlier in the visual exercise were different from that chosen with the K- means clustering algorithm. The K-means clustering exercise should therefore, be run again with this data, but with new random centroid starting values. With many runs, the cluster definitions are likely to stabilize. If the cluster definitions do not stabilize, that may be a sign that the number of clusters chosen is too high or too low. The algorithm should also be run with different values of K.

Here is the pseudo code for implementing a K-means algorithm.

Algorithm K-Means (K number of clusters, D list of data points)

1. Choose K number of random data points as initial centroids (cluster-centers)
2. Repeat till cluster-centers stabilize
   a. { *Allocate each point in D to the nearest of K centroids;*
   b. *Compute centroid for the cluster using all points in*

4.7 Selecting the number of clusters

The correct choice of the value of *k* is often ambiguous. It depends on the shapeand scale of the distribution points in a data set and the desired clustering resolution of the user. Heuristics are needed to pick the right number. One can graph the percentage of variance explained by the clusters against the number of clusters (Fig 9.10). The first clusters will add more information (explain a lot of variance), but at some point the marginal gain in variance will fall, giving a sharp angle to the graph, looking like an elbow. Beyond that elbow point, adding more clusters will not add muchincremental value. That would be the desired K.

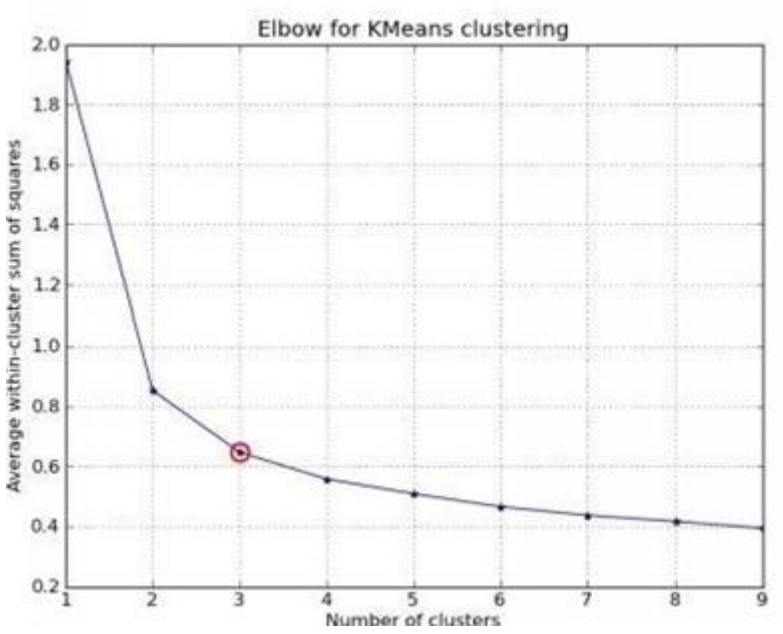


Figure 4.5: Elbow method for determining number of clusters in a data set

To engage with the data and to understand the clusters better, it is often better to start with a small number of clusters such as 2 or 3, depending upon the data set and the application domain. The number can be increased subsequently, as needed from an application point of view. This helps understand the data and the clusters progressively better.

4.8 Advantages and Disadvantages of K-Means algorithm

There are many advantages of K-Means Algorithm

1. K-Means algorithm is simple, easy to understand and easy to implement.
2. It is also efficient, in that the time taken to cluster k-means, rises linearly with the number of data points.
3. No other clustering algorithm performs better than K-Means, in general. There are a few disadvantages too:

1. The user needs to specify an initial value of K.
2. The process of finding the clusters may not converge.
3. It is not suitable for discovering clusters shapes that are not hyper- ellipsoids (or hyper-spheres).

Neural networks can also be deployed for clustering, using the appropriate objective function. The neural network will produce the appropriate cluster centroids and cluster population for each cluster.

Chapter 5: Association Rule Mining

Associate rule mining is a popular, unsupervised learning technique, used in business to help identify shopping patterns. It is also known as market basket analysis. It helps find interesting relationships (affinities) between variables (items or events). Thus, it can help cross-sell related items and increase the size of a sale.

All data used in this technique is categorical . There is no dependent variable. It uses machine learning algorithms. The fascinating "relationship between sales of diapers and beers' is how it is often explained in popular literature. This technique accepts as input the raw point-of- sale transaction data. The output produced is the description of the most frequent affinities among items. An example of an association rule would be, "A Customer who bought a flight tickets and a hotel reservation also bought a rental car plan 60 percent of the time."

5.1 Business Applications of Association Rules

In business environments a pattern or knowledge can be used for many purposes. In sales and marketing, it is used for cross-marketing and cross- selling, catalog design, e-commerce site design, online advertising optimization, product pricing, and sales/promotion configurations. This analysis can suggest not to put one item on sale at a time, and instead to create a bundle of products promoted as a package to sell other non-selling items.

In retail environments, it can be used for store design. Strongly associated items can be kept close tougher for customer convenience. Or they could be placed far from each other so that the customer has to walk the aisles and by doing so is potentially exposed to other items.

In medicine, this technique can be used for relationships between symptoms and illnesses; diagnosis and patient characteristics/treatments; genes and their functions; etc.

5.2 Representing Association Rules

A generic Association Rule is represented between a set X and Y: **XÞ Y [S%, C%]**

X, Y: products and/or services

X: Left-hand-side (LHS)

Y: Right-hand-side (RHS)

S: Support: how often **X** and **Y** go together in the dataset – i.e. P (**X** U **Y**) **C:**
Confidence: how often **Y** is found, given **X** – i.e. P (**Y** | **X**)
Example: {Hotel booking, Flight booking}Þ {Rental Car} [30%, 60%]

[Note: P (X) is the mathematical representation of the the probability or chance of X occurring in the data set.}
Computation example:
Suppose there are 1000 transactions in a data set. There are 300 occurrences of X, and 150 occurrences of (X,Y) in the data set.
Support S for **XÞ Y** will be P(X U Y) = 150/1000 = 15%.

Confidence for **XÞ Y** will be P (**Y** | **X**); or P (**X** U **Y**) / P (**X**) = 150/300 = 50%

5.3 Algorithms for Association Rule

Not all association rules are interesting and useful, only those that are strong rules and also those that occur frequently. In association rule mining, the goal is to find all rules that satisfy the user-specified *minimum support* and *minimum confidence*. The resulting sets of rules are all the same irrespective of the algorithm used, that is, given a transaction data set T, a minimum support and a minimum confidence, the set of association rules existing in T is *uniquely determined*.
Fortunately, there is a large number of algorithms that are available for generating association rules. The most popular algorithms are Apriori, Eclat, FP-Growth, along with various derivatives and hybrids of the three. All the algorithms help identify the frequent item sets, which are then converted to association rules.

5.4 Apriori Algorithm

This is the most popular algorithm used for association rule mining. The objective is to find subsets that are common to at least a minimum number of the itemsets. A frequent itemset is an itemset whose support is greater than or equal to minimum support threshold. The Apriori property is a downward closure property, which means that any subsets of a frequent itemset are also frequent itemsets. Thus, if (A,B,C,D) is a frequent itemset, then any subset such as (A,B,C) or (B,D) are also frequent itemsets.
It uses a bottom-up approach; and the size of frequent subsets is gradually increased, from

one-item subsets to two-item subsets, then three-itemsubsets, and so on. Groups of candidates at each level are tested against the data for minimum support.

5.5 Association rules exercise

Here are a dozen sales transactions. There are six products being sold: Milk, Bread, Butter, Eggs, Cookies, and Ketchup. Transaction#1 sold Milk, Eggs, Bread and Butter. Transaction#2 sold Milk, Butter, Egg & Ketchup. And so on. The objective is to use this transaction data to find affinities between products, i.e. which products sell together often.

The support level will be set at 33 percent; the confidence level will be set at 50 percent. That means that we have decided to consider rules from only those itemsets that occur at least 33 percent of the time in the total set of transactions. Confidence level means that within those itemsets, the rules of the form X → Y should be such that there is at least 50 percent chance of Y occurring based on X occurring.

	Transactions List			
1	Milk	Egg	Bread	Butter
2	Milk	Butter	Egg	Ketchup
3	Bread	Butter	Ketchup	
4	Milk	Bread	Butter	
5	Bread	Butter	Cookies	
6	Milk	Bread	Butter	Cookies
7	Milk	Cookies		
8	Milk	Bread	Butter	
9	Bread	Butter	Egg	Cookies
10	Milk	Butter	Bread	
11	Milk	Bread	Butter	

12	Milk	Bread	Cookies	Ketchup

First step is to compute 1-item Itemsets.i.e. How often does any product individually sell.

1-item Sets	Frequency
Milk	9
Bread	10
Butter	10
Egg	3
Ketchup	3
Cookies	5

Thus, Milk sells in 9 out of 12 transactions. Bread sells in 10 out of 12 transactions. And so on. At every point, there is an opportunity to select itemsets of interest, and thus further analysis. Other itemsets that occur very infrequently may be removed. If itemsets that occur 4 or more times out of 12 are selected, that corresponds to meeting a minimum support level of 33 percent (4 out of 12). Only 4 items make the cut. The frequent items that meet the support level of 33 percent are:

Frequent 1-item Sets	Freq
Milk	9
Bread	10

Butter	10
Cookies	5

The next step is to go for the next level of itemsets using items selected earlier: 2-item itemsets.

2-item Sets	Frequency
Milk, Bread	7
Milk, Butter	7
Milk, Cookies	3
Bread, Butter	9
Butter, Cookies	3
Bread, Cookies	4

Thus (Milk, Bread) sell 7 times out of 12. (Milk, Butter) sell together 7 times, (Bread, Butter sell) together 9 times, and (Bread, Cookies) sell 4 times.

However only four of these transactions meet the minimum support level of 33%.

2-item Sets	**Freq**
Milk, Bread	7
Milk, Butter	7
Bread, Butter	9
Bread, Cookies	4

The next step is to list the next higher level of itemsets: 3-item itemsets.

3-item Sets	Freq
Milk, Bread, Butter	6
Milk, Bread, Cookies	1
Bread, Butter, Cookies	3

Thus (Milk, Bread, Butter) sell 6 times out of 12. (Bread, Butter, Cookies) sell 3 times out of 12. One one 3-item itemset meets the minimum support requirements.

3-item Sets	Freq
Milk, Bread, Butter	6

There is no room to create a 4-item itemset for this support level.

5.6 Creating Association Rules

The most interesting and complex rules at higher size itemsets start top-down with the most frequent itemsets of higher size-numbers. Association rules are created that meet the support level (>33%) and confidence levels (> 50%).

The highest level itemset that meets the support requirements is the three-item itemset. The following itemset has a support level of 50% (6 out of 12).

Milk, Bread, Butter	6

This itemset could lead to multiple candidate Association rules. Start with the following rule: (Bread, Butter) Milk.
There are a total of total 12 transactions.

X (in this case Bread, Butter) occurs 9 times;

X,Y (in this case Bread, Butter, Milk) occurs 6 times.

The support level for this rule is 6/12 = 50%. The confidence level for this rule is 6/9 = 67%. This rule meets our thresholds for support (>33%) and confidence (>50%).

Thus, the first valid Association rule from this data is: (**Bread, Butter)Milk {S=50%, C=67%}**.

In exactly the same way, other rules can be considered for their validity.

Consider the rule: (Milk, Bread) Butter. Out of total 12 transactions, (Milk, Bread) occur 7 times; and (Milk, Bread, Butter) occurs 6 times.

The support level for this rule is 6/12 = 50%. The confidence level for this rule is 6/7 = 84%. This rule meets our thresholds for support (>33%) and confidence (>50%).

Thus, the second valid Association rule from this data is (**Milk, Bread)Butter {S=50%, C=67%}**.

Consider the rule (Milk, Butter)Bread. Out of total 12 transactions (Milk,Butter) occurs 7 times while (Milk, Butter, Bread) occur 6 times.
The support level for this rule is 6/12 = 50%. The confidence level for this rule is 6/7 = 84%. This rule meets our thresholds for support (>33%) and confidence (>50%).
Thus, the next valid Association rule is: **Milk,Butter Bread {S=50%, C=84%}**.
Thus, there were only three possible rules at the 3-item itemset level, and all were found to be valid.
One can get to the next lower level and generate association rules at the 2- item itemset level.

Consider the rule Milk Bread.Out of total 12 transactions Milk occurs 9 times while (Milk, Bread) occur 7 times.
The support level for this rule is 7/12 = 58%. The confidence level for this rule is 7/9 = 78%.

This rule meets our thresholds for support (>33%) and confidence (>50%).
Thus, the next valid Association rule is:

Milk -> Bread {58%, 77%}.

Many such rules could be derived if needed.

Not all such association rules are interesting. The client may be interested in only the top few rules that they want to implement. The number of association rules depends upon business need. Implementing every rule in business will require some cost and effort, with some potential of gains. The strongest of rules, with the higher support and confidence rates, should be used first, and the others should be progressively implemented later.

MODULE 5

Chapter 1: Text Mining

Text mining is the art and science of discovering knowledge, insights and patterns from an organized collection of textual databases. Textual mining can help with frequency analysis of important terms, and their semantic relationships.

Text is an important part of the growing data in the world. Social media technologies have enabled users to become producers of text and images and other kinds of information. Text mining can be applied to large-scale social media data for gathering preferences, and measuring emotional sentiments. It can also be applied to societal, organizational and individual scales.

1.1 Text Mining Applications

Text mining is a useful tool in the hands of chief knowledge officers to extract knowledge relevant to an organization. Text mining can be used across industry sectors and application areas, including decision support, sentiment analysis, fraud detection, survey analysis, and many more.

1. *Marketing*: The voice of the customer can be captured in its native and raw format and then analyzed for customer preferences and complaints.
    1. Social personas are a clustering technique to develop customer segments of interest. Consumer input from social media sources, such as reviews, blogs, and tweets, contain numerous leading indicators that can be used towards anticipating and predicting consumer behavior.
    2. A 'listening platform' is a text mining application, that in real time, gathers social media, blogs, and other textual feedback, and filters out the chatter to extract true consumer sentiment. The insights can lead to more effective product marketing and better customer service.

2. The customer call center conversations and records can be analyzed for patterns of customer complaints. Decision trees can organize this data to create decision choices that could help with product management activities and to become proactive in avoiding those complaints.

3. *Business operations*: Many aspects of business functioning can be accurately gauged from analyzing text./
    1. Social network analysis and text mining can be applied to emails, blogs, social media and other data to measure the emotional states and the mood of employee populations. Sentiment analysis can reveal early signs of employee dissatisfaction which can then can be proactively managed.

2. Studying people as emotional investors and using text analysis of the social Internet to measure mass psychology can help in obtaining superior investment returns.

3. *Legal*: In legal applications, lawyers and paralegals can more easily search case histories and laws for relevant documents in a particular case to improve their chances of winning.
    1. Text mining is also embedded in e-discovery platforms that help in minimizing risk in the process of sharing legally mandated documents.
    2. Case histories, testimonies, and client meeting notes can reveal additional information, such as morbidities in a healthcare situation that can help better predict high-cost injuries and preventcosts.

4. Governance and Politics: Governments can be overturned based on a tweet originating from a self-immolating fruit-vendor inTunisia.
    1. Social network analysis and text mining of large-scale social media data can be used for measuring the emotional states and the mood of constituent populations. Micro-targeting constituents with specific messages gleaned from social media analysis can be a more efficient use of resources when fighting democratic elections.
    2. In geopolitical security, internet chatter can be processed for real- time information and to connect the dots on any emerging threats.
    3. In academic, research streams could be meta-analyzed for underlying research trends.

1.2 Text Mining Process

Text Mining is a rapidly evolving area of research. As the amount of social media and other text data grows, there is need for efficient abstraction and categorization of meaningful information from the text.

The first level of analysis is identifying frequent words. This creates a bag of important words. Texts – documents or smaller messages – can then be ranked on how they match to a particular bag-of-words. However, there are challenges with this approach. For example, the words may be spelled a little differently. Or there may be different words with similar meanings.

The next level is at the level of identifying meaningful phrases from words. Thus ‘ice’ and ‘cream’ will be two different key words that often come together. However, there is a more meaningful phrase by combining the two words into ‘ice cream’. There might be similarly meaningful phrases like ‘Apple Pie’.

The next higher level is that of Topics. Multiple phrases could be combined into Topic area. Thus the two phrases above could be put into a common basket, and this bucket could be

called 'Desserts'.

Text mining is a semi-automated process. Text data needs to be gathered, structured, and then mined, in a 3-step process (Figure 11.1)

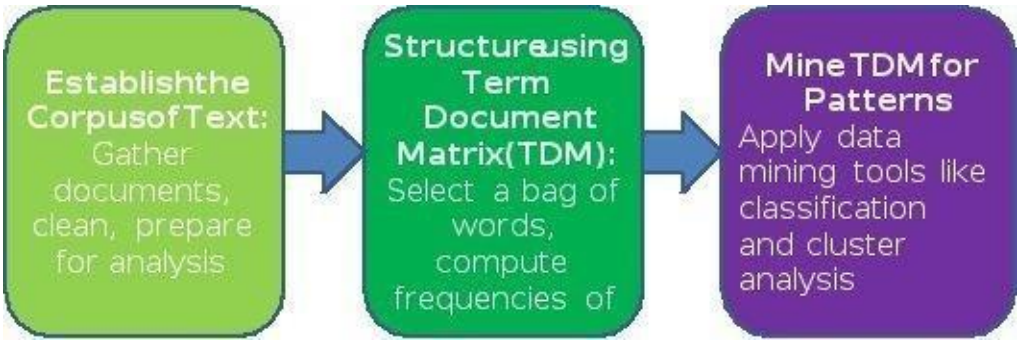


Figure 1.1: Text Mining Architecture

1. The text and documents are first gathered into a corpus, and organized.
2. The corpus is then analyzed for structure. The result is a matrix mapping important terms to source documents.
3. The structured data is then analyzed for word structures, sequences, and frequency.

1.3 Term Document Matrix

This is the heart of the structuring process. Free flowing text can be transformed into numeric data in a TDM, which can then be mined using regular data mining techniques.

1. There are several efficient techniques for identifying key terms from a text. There are less efficient techniques available for creating topics out of them. For the purpose of this discussion, one could call key words, phrases or topics as a term of interest. This approach measures the frequencies of select important terms occurring in each document. This creates a t x d Term–by–Document Matrix (TDM) where t is the number of terms and d is the number of documents (Table 11.1).
2. Creating a TDM requires making choices of which terms to include. The terms chosen should reflect the stated purpose of the text mining exercise. The list of terms should be as extensive as needed, but should not include unnecessary stuff that will serve to confuse the analysis, or slow the computation.

	Term Document Matrix				
Document / Terms	investment	Profit	happy	Success	…
Doc 1	10	4	3	4	
Doc 2	7	2	2		
Doc 3			2	6	
Doc 4	1	5	3		
Doc 5		6		2	
Doc 6	4		2		
…					

Table 1.1: Term-Document Matrix

Here are some considerations in creating a TDM.

1. A large collection of documents mapped to a large bag of words will likely lead to a very sparse matrix if they have few common words. Reducing dimensionality of data will help improve the speed of analysis and meaningfulness of the results. Synonyms, or terms will similar meaning, should be combined and should be counted together, as a common term. This would help reduce the number of distinct terms of words or 'tokens'.
2. Data should be cleaned for spelling errors. Common spelling errors should be ignored and the terms should be combined. Uppercase- lowercase terms should also be combined.
3. When many variants of the same term are used, just the stem of theword would be used to reduce the number of terms. For instance, terms like customer order, ordering, order data, should be combined into a single token word, called 'Order'.
4. On the other side, homonyms (terms with the same spelling but different meanings) should be counted separately. This would enhance the quality of analysis. For example, the term order can mean a customer order, or the ranking of certain choices. These two should be treated separately. "The boss ordered that the customer orders data analysis be presented in chronological order'. This statement shows three different meanings for the word 'order'. Thus, there will be a need for a manual review of the TD matrix.
5. Terms with very few occurrences in very few documents should be eliminated from the matrix. This would help increase the density of the matrix and the quality of analysis.
6. The measures in each cell of the matrix could be one of several possibilities. It could be a simple count of the number of occurrences of each term in a document. It could also be the log of that number. It could be the fraction number computed by dividing the frequency count by the total number of words in the document. Or there may be binary valuesin the matrix to represent whether a term is mentioned or not. The choice of value in the cells will depend upon the purpose of the textanalysis.

At the end of this analysis and cleansing, a well-formed, densely populated, rectangular, TDM will be ready for analysis. The TDM could be mined using all the available data mining techniques.

1.4 Mining the TDM

The TDM can be mined to extract patterns/knowledge. A variety of techniques could be applied to the TDM to extract new knowledge.

Predictors of desirable terms could be discovered through predictive techniques, such as regression analysis. Suppose the word profit is a desirable word in a document. The number of occurrences of the word profit in a document could be regressed against many other terms in the TDM. The relative strengths of the coefficients of various predictor variables would show the relative impact of those terms on creating a profitdiscussion.

Predicting the chances of a document being liked is another form of analysis. For example, important speeches made by the CEO or the CFO to investors could be evaluated for quality. If the classification of those documents (such as good or poor speeches) was available, then the terms of TDM could be used to predict the speech class. A decision tree could be constructed that makes a simple tree with a few decision points that predicts the success of a speech 80 percent of the time. This tree could be trained with more data to become better over time.

Clustering techniques can help categorize documents by common profile. For example, documents containing the words investment and profit more often could be bundled together. Similarly, documents containing the words, customer orders and marketing, more often could be bundled together. Thus, a few strongly demarcated bundles could capture the essence of the entire TDM. These bundles could thus help with further processing, such as handing over select documents to others for legal discovery.

Association rule analysis could show relationships of coexistence. Thus, one could say that the words, tasty and sweet, occur together often (say 5 percent of the time); and further, when these two words are present, 70 percent of the time, the word happy, is also present in the document.

1.5 Comparing Text Mining and Data Mining

Text Mining is a form of data mining. There are many common elements between Text and Data Mining. However, there are some key differences (Table 1.2). The key difference is that text mining requires conversion of text data into frequency data, before data mining techniques can beapplied.

Dimension	**Text Mining**	**Data Mining**
Nature of data	Unstructured data: Words, phrases, sentences	Numbers; alphabetical and logical values
Language used	Many languages and dialects used in the world; many languages are extinct, new documents are discovered	Similar numerical systems across the world
Clarity and precision	Sentences can be ambiguous; sentiment may contradict the words	Numbers are precise.
Consistency	Different parts of the text can contradict each other	Different parts of data can be inconsistent, thus, requiring statistical significance analysis
Sentiment	Text may present a clear and consistent or mixed sentiment, across a continuum. Spokenwords adds further sentiment	Not applicable
Quality	Spelling errors. Differing values of proper nouns such as names. Varying quality of language translation	Issues with missing values, outliers, etc
Nature of Analysis	Keyword based search; co- existence of themes; Sentiment Mining	A full wide range of statistical and machine learning analysis for relationship and differences

Table 1.2: Comparing Text Mining and Data Mining

1.6 Text Mining Best Practices

Many of the best practices that apply to the use of data mining techniques will also apply to text mining.

1. The first and most important practice is to ask the right question. A good question is one which gives an answer and would lead to large payoffs for the organization. The purpose and the key question will define how and at what levels of granularity the TDM would be made. For example, TDM defined for simpler searches would be different from those used for complex semantic analysis or network analysis.
2. A second important practice is to be creative and open in proposing imaginative hypotheses for the solution. Thinking outside the box is important, both in the quality of the proposed solution as well as in finding the high quality data sets required to test the hypothesized solution. For example, a TDM of consumer sentiment data should be combined with customer order data in order to develop a comprehensive view of customer behavior. It's important to assemble a team that has a healthy mix of technical and business skills.
3. Another important element is to pursue the problem iteratively. Too much data can overwhelm the infrastructure and also befuddle the mind. It is better to divide and conquer the problem with a simpler TDM, with fewer terms and fewer documents and data sources. Expand as needed, in an iterative sequence of steps. In the future, add new terms to help improve predictive accuracy.
4. A variety of data mining tools should be used to test the relationships in the TDM. Different decision tree algorithms could be run alongside cluster analysis and other techniques. Triangulating the findings with multiple techniques, and many what-if scenarios, helps build confidence in the solution. Test the solution in many ways before committing to deploy it.

Chapter 2: Web Mining

Web mining is the art and science of discovering patterns and insights from the World-wide web so as to improve it. The world-wide web is at the heart of the digital revolution. More data is posted on the web every day than was there on the whole web just 20 years ago. Billions of users are using it every day for a variety of purposes. The web is used for electronic commerce, business communication, and many other applications. Web mining analyzes data from the web and helps find insights that could optimize the web content and improve the user experience. Data for web mining is collected via Web crawlers, web logs, and other means.

Here are some characteristics of optimized websites:

1. *Appearance*: Aesthetic design. Well-formatted content, easy to scan and navigate. Good color contrasts.
2. *Content*: Well-planned information architecture with useful content. Fresh content. Search-engine optimized. Links to other goodsites.
3. *Functionality*: Accessible to all authorized users. Fast loading times. Usable forms. Mobile enabled.

This type of content and its structure is of interest to ensure the web is easy to use. The analysis of web usage provides feedback on the web content, and also the consumer's browsing habits. This data can be of immense use for commercial advertising, and even for social engineering.

The web could be analyzed for its structure as well as content. The usage pattern of web pages could also be analyzed. Depending upon objectives, web mining can be divided into three different types: Web usage mining, Web content mining and Web structure mining (Figure 2.1).

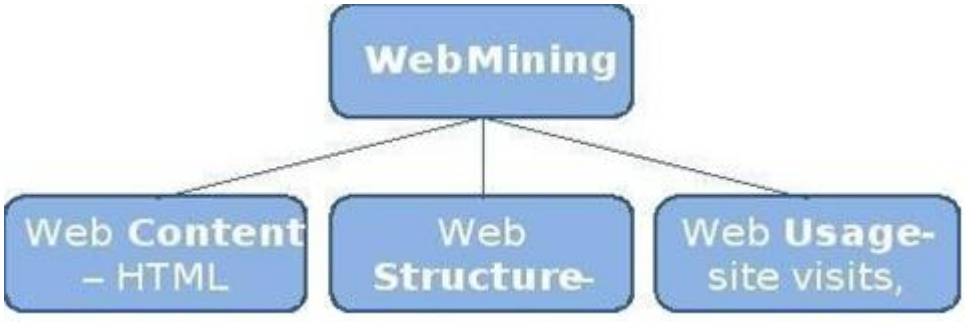


Figure: 2.1 Web Mining structure

2.1 Web content mining

A website is designed in the form of pages with a distinct URL (universal resource locator). A large website may contain thousands of pages. These pages and their content is managed using specialized software systems called Content Management Systems. Every page can have text, graphics, audio, video, forms, applications, and more kinds of content including user generated content.

The websites keep a record of all requests received for its page/URLs, including the requester information using 'cookies'. The log of these requests could be analyzed to gauge the popularity of those pages among different segments of the population. The text and application content on the pages could be analyzed for its usage by visit counts. The pages on a website themselves could be analyzed for quality of content that attracts most users. Thus the unwanted or unpopular pages could be weeded out, or they can be transformed with different content and style. Similarly, more resources could be assigned to keep the more popular pages more fresh and inviting.

2.2 Web structure mining

The Web works through a system of hyperlinks using the hypertext protocol (http). Any page can create a hyperlink to any other page, it can be linked to by another page. The intertwined or self-referral nature of web lends itself to some unique network analytical algorithms. The structure of Web pages could also be analyzed to examine the pattern of hyperlinks among pages. There are two basic strategic models for successful websites: Hubs and Authorities.

1. *Hubs*: These are pages with a large number of interesting links. They serve as a hub, or a gathering point, where people visit to access a variety of information. Media sites like Yahoo.com, or government sites would serve that purpose. More focused sites like Traveladvisor.com and yelp.com could aspire to becoming hubs for new emergingareas.
2. *Authorities*: Ultimately, people would gravitate towards pages that provide the most complete and authoritative information on a particular subject. This could be factual information, news, advice, user reviews etc. These websites would have the most number of inbound links from other websites. Thus Mayoclinic.com would serve as an authoritative page for expert medical opinion. NYtimes.com would serve as an authoritative page for daily news.

2.3 Web usage mining

As a user clicks anywhere on a webpage or application, the action is recorded by many entities in many locations. The browser at the client machine will record the click, and the web server providing the content would also make a record of the pages served and the user activity on those pages. The entities between the client and the server, such as the router, proxy server, or ad server, too would record that click.

The goal of web usage mining is to extract useful information and patterns from data generated through Web page visits and transactions. The activity data comes from data stored

in server access logs, referrer logs, agent logs, and client-side cookies. The user characteristics and usage profiles are also gathered directly, or indirectly, through syndicated data. Further, metadata, such as page attributes, content attributes, and usage data are also gathered.

The web content could be analyzed at multiple levels (Figure 2.2).

1. The *server side analysis* would show the relative popularity of the web pages accessed. Those websites could be hubs and authorities.
2. The *client side analysis* could focus on the usage pattern or the actual content consumed and created by users.
    1. Usage pattern could be analyzed using 'clickstream' analysis, i.e. analyzing web activity for patterns of sequence of clicks, and the location and duration of visits on websites. Clickstream analysis can be useful for web activity analysis, software testing, market research, and analyzing employee productivity.
    2. Textual information accessed on the pages retrieved by users could be analyzed using text mining techniques. The text would be gathered and structured using the bag-of-words technique to build a Term-document matrix. This matrix could then be mined using cluster analysis and association rules for patterns such as popular topics, user segmentation, and sentiment analysis.

Figure: 2.2 Web Usage Mining architecture

Web usage mining has many business applications. It can help predict user behavior based on previously learned rules and users' profiles, and can help determine lifetime value of clients. It can also help design cross-marketing strategies across products, by observing association rules among the pages on the website. Web usage can help evaluate promotional campaigns and see if the users were attracted to the website and used the pages relevant to the campaign. Web usage mining could be used to present dynamic information to users based on their interests and profiles. This includes targeted online ads and coupons at user groups based on user access patterns.

2.4 Web Mining Algorithms

Hyperlink-Induced Topic Search (HITS) is a link analysis algorithm that rates web pages as being hubs or authorities. Many other HITS-based algorithms have also been published. The most famous and powerful of these algorithms is the PageRank algorithm. Invented by

Google co-founder Larry Page, this algorithm is used by Google to organize the results of its search function. This algorithm helps determine the relative importance of any particular web page by counting the number and quality of links to a page. The websites with more number of links, and/or more links from higher-quality websites, will be ranked higher. It works in a similar way as determining the status of a person in a society of people. Those with relations to more people and/or relations to people of higher status will be accorded a higher status.

PageRank is the algorithm that helps determine the order of pages listed upon a Google Search query. The original PageRank algorithm formuation has been updated in many ways and the latest algorithm is kept a secret so other websites cannot take advantage of the algorithm and manipulate their website according to it. However, there are many standard elements that remain unchanged. These elements lead to the principles for a good website. This process is also called Search Engine Optimization (SEO).

Chapter 3: Web Mining

Naïve Bayes technique is a supervised machine learning technique that that uses probability theory based analysis.

What is Naive Bayes Algorithm?

Naive Bayes Algorithm is one of the popular classification machine learning algorithms that helps to classify the data based upon the conditional probability values computation. It implements the Bayes theorem for the computation and used class levels represented as feature values or vectors of predictors for classification. Naive Bayes Algorithm is a fast algorithm for classification problems. This algorithm is a good fit for real-time prediction, multi-class prediction, recommendation system, text classification, and sentiment analysis use cases. Naive Bayes Algorithm can be built using Gaussian, Multinomial and Bernoulli distribution. This algorithm is scalable and easy to implement for the large data set.

It helps to calculate the posterior probability P(c|x) using the prior probability of class P(c), the prior probability of predictor P(x) and the probability of predictor given class, also called as likelihood P(x|c).

The formula or equation to calculate posterior probability is:

- $P(c|x) = (P(x|c) * P(c)) / P(x)$

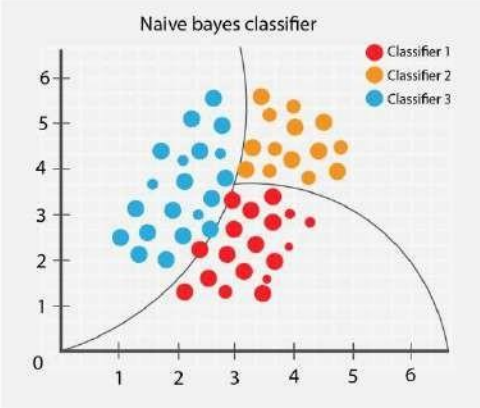
- It is a supervised learning technique that uses probability-theory-based analysis.
- It is a machine learning technique that computes the probabilities of an instances belonging to each one of many target classes, given the prior probabilities of classification using individual factors.
- Naïve Bayes technique is used often in classifying text doucments into one multiple predefined categories.

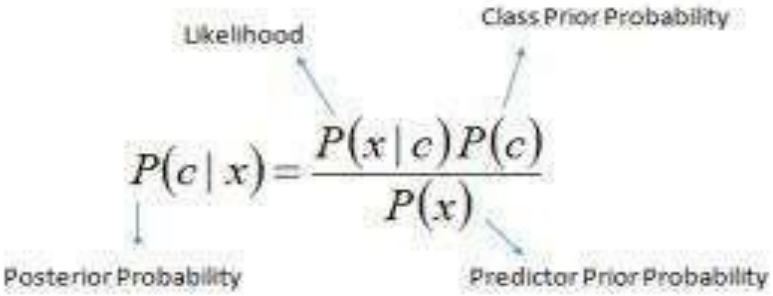
In machine learning, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

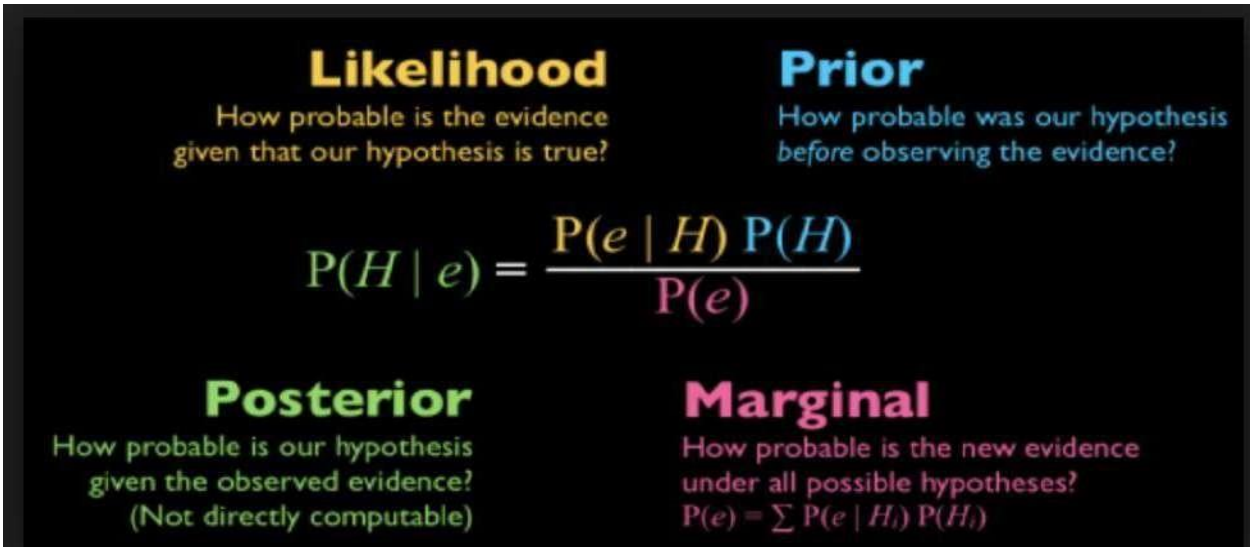
using Bayesian probability terminology, the above equation can be written as

$$\text{Posterior} = \frac{\text{prior x likelihood}}{\text{evidence}}$$





$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$



• It is machine learning technique that computes the probabilities of an instance of belonging to each of many target classes, given the prior probabilities of classification using individual factors.

Advantanges of Naïve Bayes

- NB logic is simple and so is the NB posterior probability computation.
- Conditional probabilities can be computed for discrete data and for probabilistic distributions.
- A variety of methods exist for modelling the conditional distributions of the X variables, including normal, lognormal, gamma, and Poisson.

advantage:

- The naive Bayesian model originated from classical mathematical theory and has a solid mathematical foundation and stable classification efficiency.
- It has a higher speed for large numbers of training and queries.Even with very large training sets, there is usually only a relatively small number of features for each project, and the training and classification of the project is only a mathematical operation of the feature probability;
- It works well for small-scale data, can handle multi-category tasks, and is suitable for incremental training (that is, it can train new samples in real time);
- Less sensitive to missing data, the algorithm is also relatively simple, often used for text classification;
- Naïve Bayes explains the results easily.

Disadvantages:

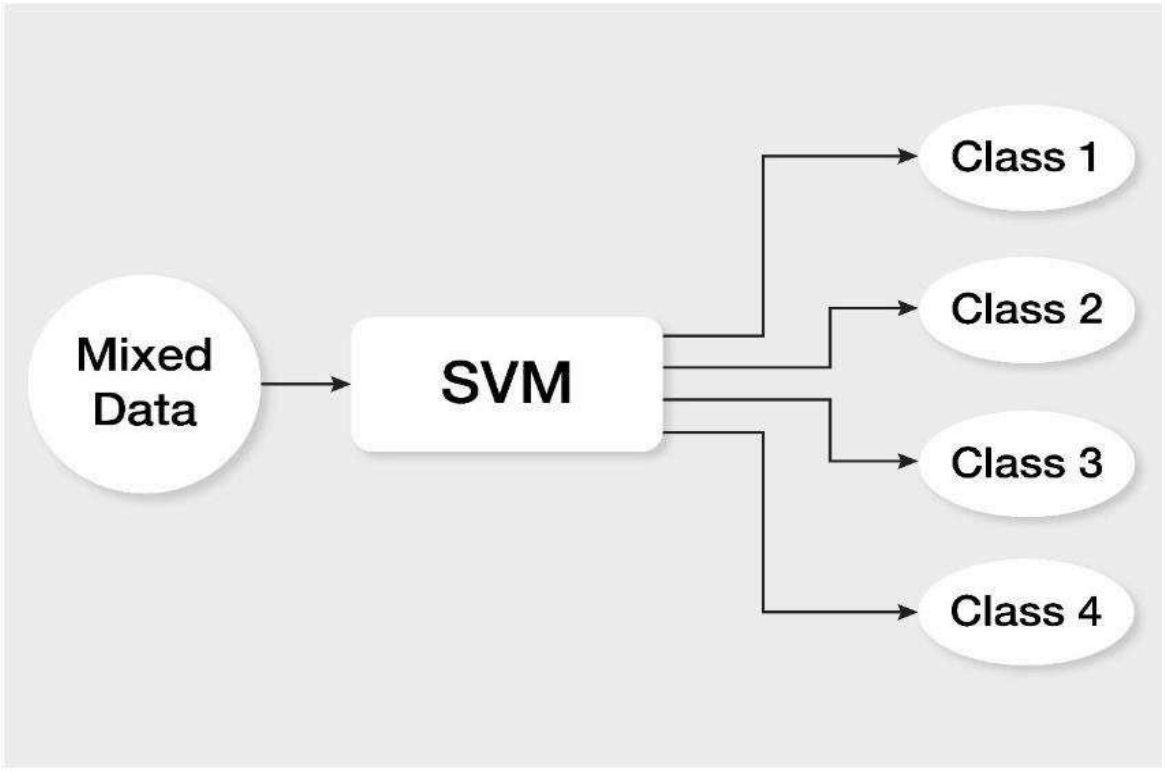
- Need to calculate the prior probability;
- There is an error rate in the classification decision;
- Very sensitive to the form of input data;
- due to**The assumption of sample attribute independence is used, so if the sample attributes are related, the effect is not good.**

Chapter 3: Support Vector Machine

• "Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges.

• However, it is mostly used in classification problems.

• In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.

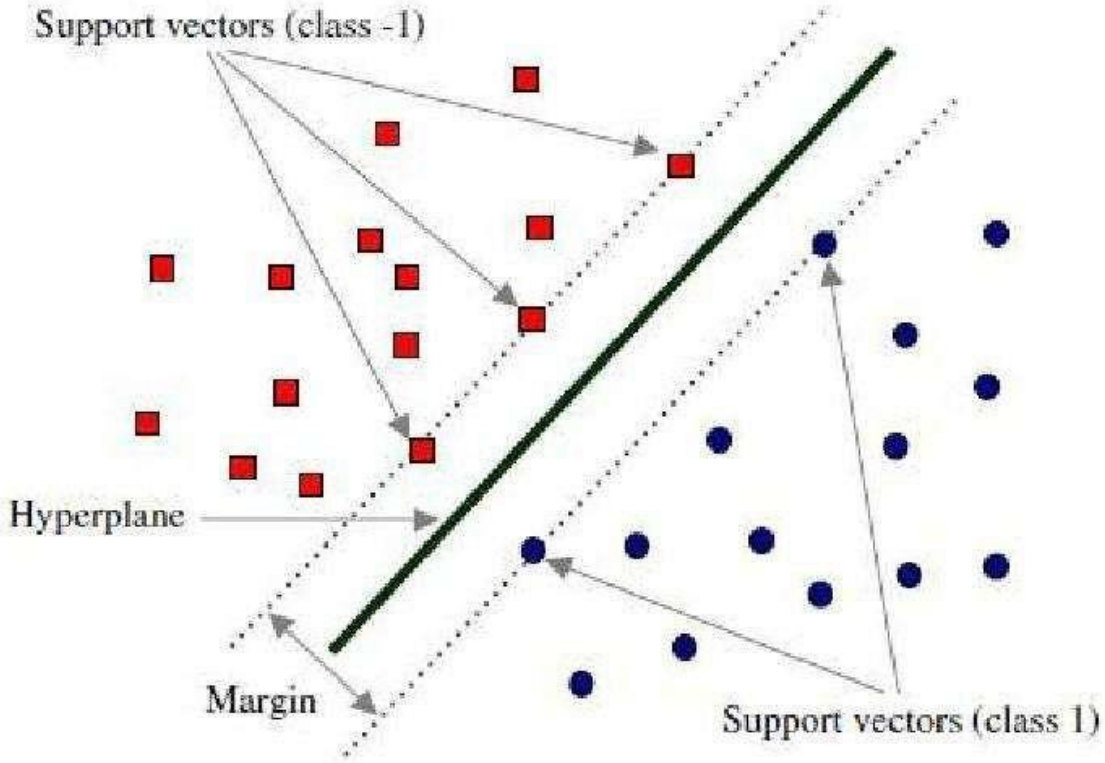


SVM classifier

- **Hyperplanes** are decision boundaries that help classify the data points.
- Data points falling on either side of the **hyperplane** can be attributed to different classes.
- In simple term, it is the ability of your **machine learning** model to correctly differentiate/separate/classify between different groups of data.

• Then, we perform classification by finding the hyper-plane that differentiate the two classes very well (look at the below snapshot).

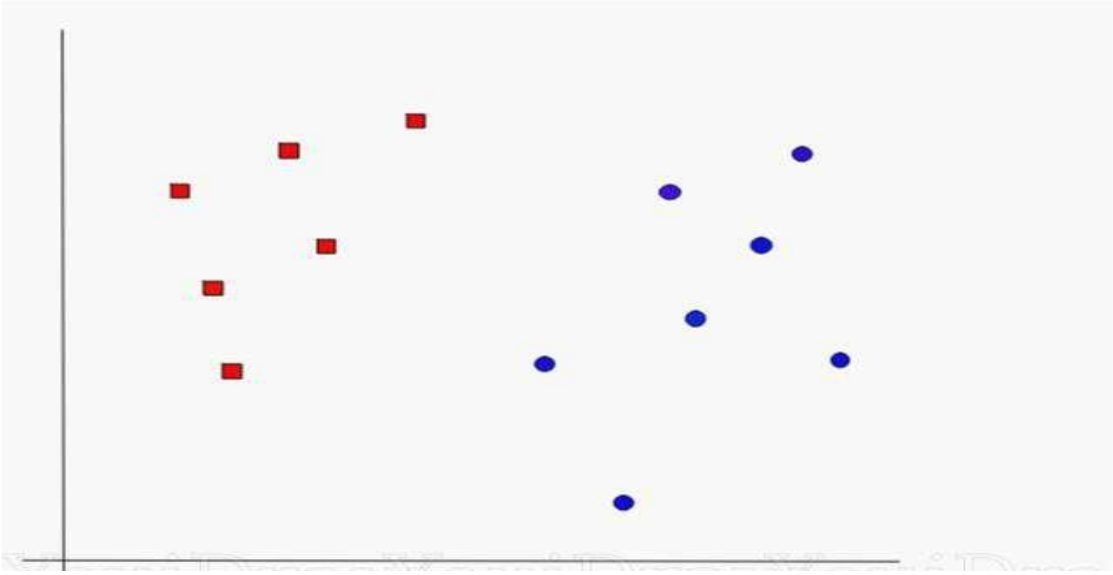
What is Support Vector Machine?

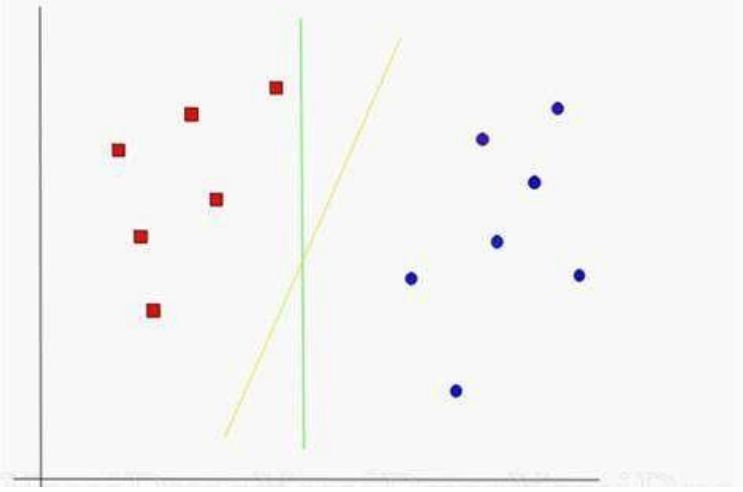


THEORY

At first approximation what SVMs do is to find a separating line(or hyperplane) between data of two classes. SVM is an algorithm that takes the data as an input and outputs a line that separates those classes if possible.

Lets begin with a problem. Suppose you have a dataset as shown below and you need to classify the red rectangles from the blue ellipses(let's say positives from the negatives). So your task is to find an ideal line that separates this dataset in two classes (say red and blue).





1. Kernel achieve this by learning from instances.
2. They remember each training example and associate a weight representing its relevance to the achievement of the objective this could be called as Instance Based Learning.
3. There are several types of support vector models including linear, Polynomial, RBF, and Sigmoid.
4. In kernel parlance we assign high weights to the abnormal situations and very low weight to the norm
5. SVMs have evolved to be more flexible and be able to tolerate some amount of misclassifications.
6. The margin of separation between categories is thus a „Soft Margine" as against a hard margin.

ADVANTAGES OF SVM

- They work well even when the number of features is much larger than the number of instances.
- It can work on datasets with huge feature space, such is the case in spam filtering, where a large number of words are the potential signifiers of a message being spam.
- Even when the optimal decision boundary is a non linear curve, the SVM transforms the variables to create new dimensions such that the representations of the classifier is a linear function of those transformed dimensions of the data.
- SVMs are conceptually easy to understand.
- They create an easy to understand linear classifier. By working on only a subset of relevant data, they are computationally efficient.
- SVMs are now available with almost all data analytics toolsets.

DISADVANTAGES OF SVM

TWO MAJOR CONSTRAINTS ARE

a) It works well only with real numbers i.e all the data points in all the dimensions must be defined by numeric values only.

b) It works only binary classification problems.

1. Training of SVMs is an inefficient and time consuming process, when the data is large.
2. It does not work well when there is much noise in the data, thus has to compute soft margins.
3. The SVMs will also not provide a probability estimate of classification.

How does it work?

Thumb rule to identify the right hyper-plane

• Select the hyper-plane which segregates the two classes better

• Maximizing the distances between nearest data point (either class) and hyper-plane. This distance is called as **Margin**.

SVM Model

• $f(x) = W.X + b$

W is the normal to the line, X is input vector and b the bias

• W is known as the weight vector

•

Advantages of SVM

• The main strength of SVM is that they work well even when the number of SVM features is much larger than the number of instances.

• It can work on datasets with huge feature space, such is the case in spam filtering, where a large number of words are the potential signifiers of a message being spam.

• Even when the optimal decision boundary is a nonlinear curve, the SVM transforms the variables to create new dimensions such that the representation of the classifier is a linear function of those transformed dimensions of the data.

• SVMs are conceptually easy to understand. They create an easy-to-understand linear classifier. By working on only a subset of relevant data,. they are computationally efficient. SVMs are now available with almost all data analytics toolsets.

Disadvantages of SVM

• The SVM technique has two major constraints

– It works well only with real numbers, i.e., all the data points in all the dimensions must be defined by numeric values only,

– It works only with binary classification problems. One can make a series of cascaded SVMs to get around this constraint.

• Training the SVMs is an inefficient and time consuming process, when the data is large.

• It does not work well when there is much noise in the data, and thus has to compute soft margins.

• The SVMs will also not provide a probability estimate of classification, i.e., the confidence level for classifying an instance.

Chapter 3: Social Network Analysis.

- Is a graphical representation of relationships among people and or entities.
- It is the art and science of discovering patterns of interaction and influence within the participants in a network.
- Participants could be people ,organizations, machines, concepts or any other kind of entities
- An ideal application of social network analysis will discover essential characteristics of a network including its central nodes and its subnet work structure.
- Subnetworks are clusters of nodes, where the within subnetwork connections are stronger than the connections with the nodes outside the subnetwork.

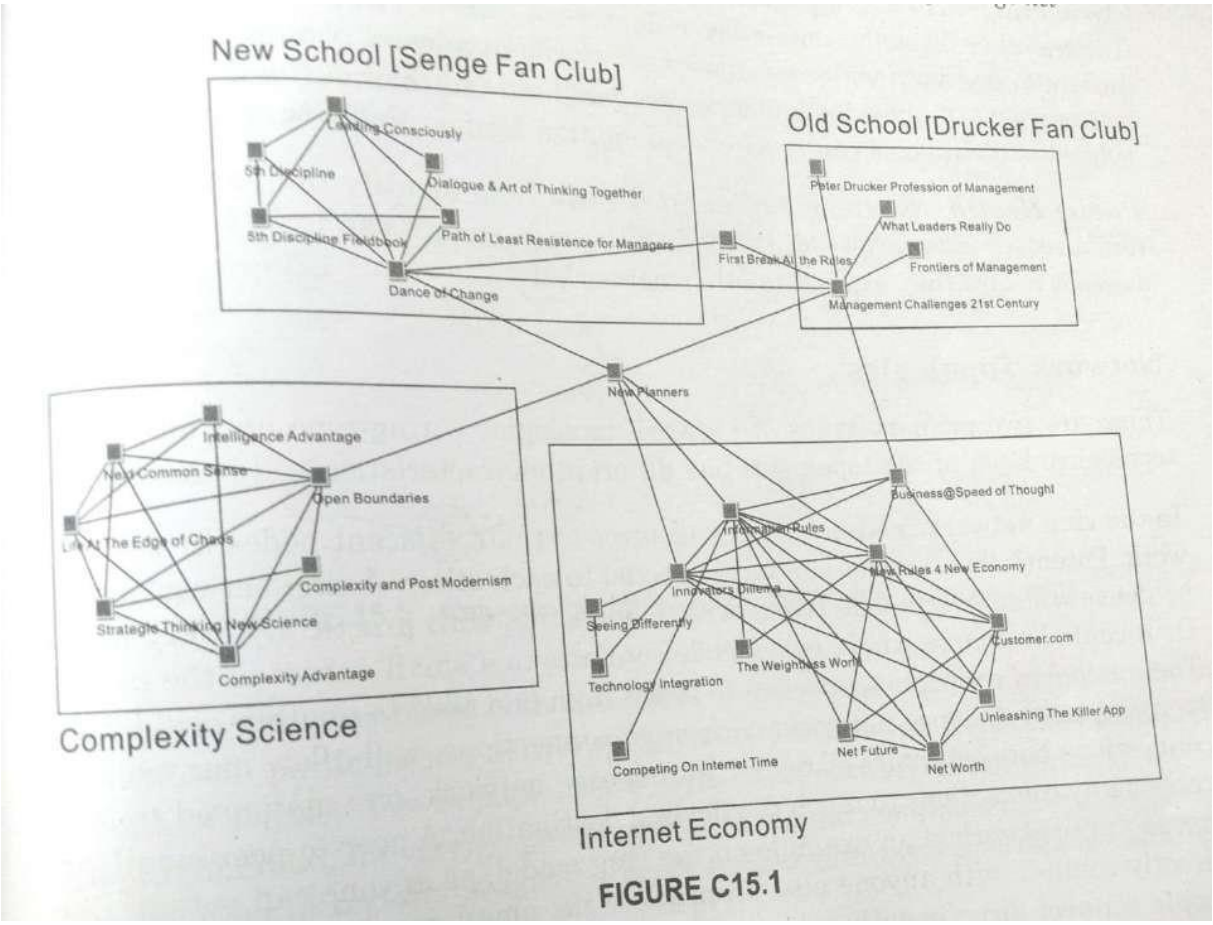
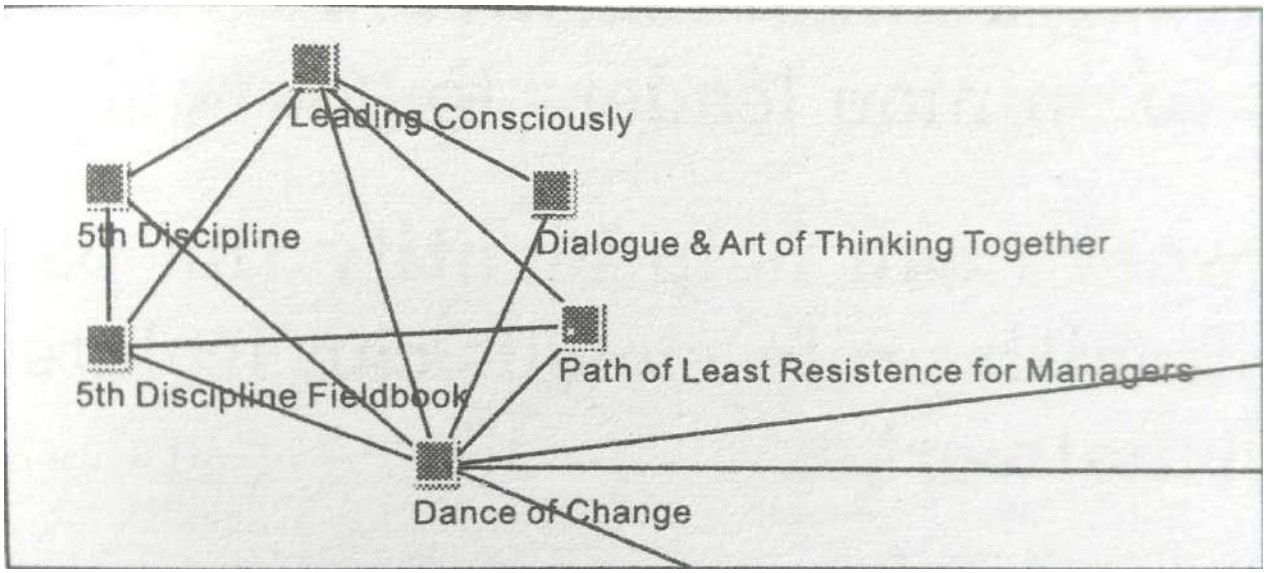
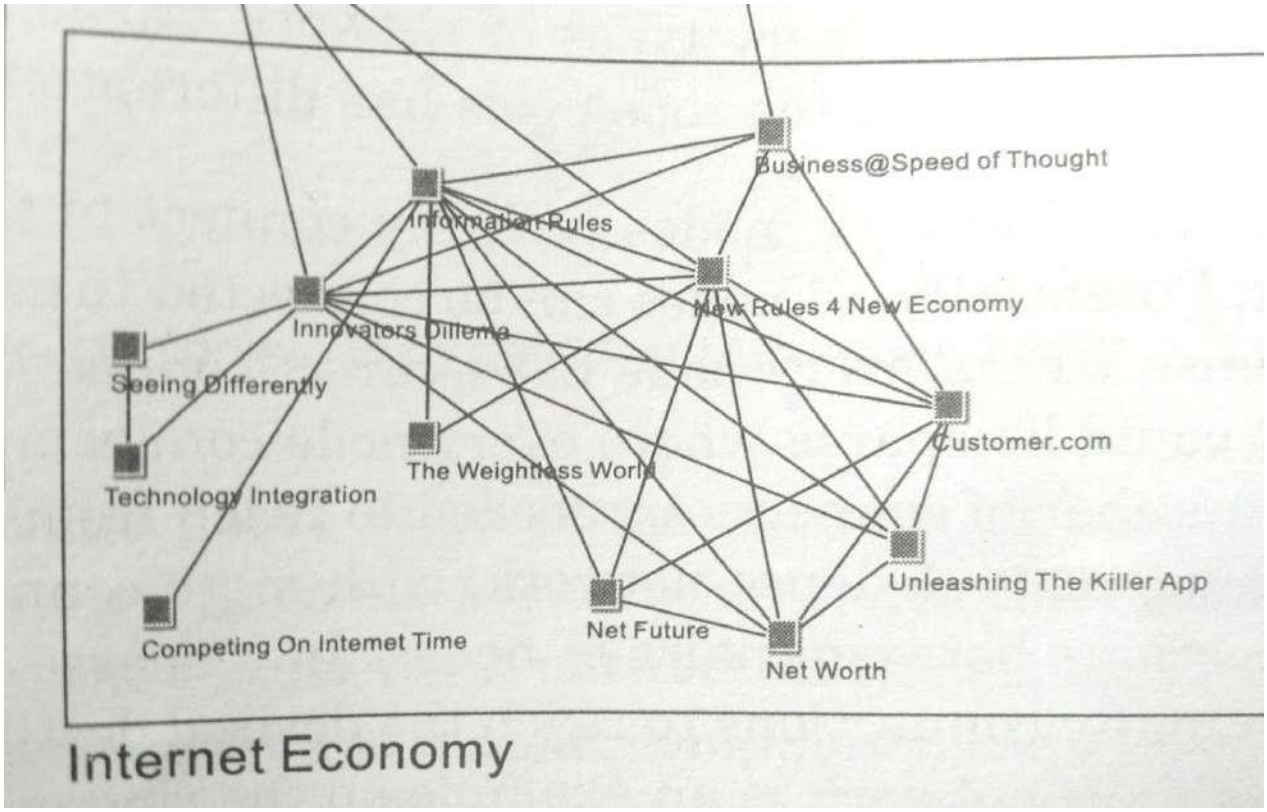


FIGURE C15.1





APPLICATIONS

- **Self – awareness:** Visualizing his/her social network can help a person organize their relationships and support network.

- **Communities:** Social Network Analysis can help identification, construction, and strengthening of networks within communities to build wellness, comfort and resilience.
- **Marketing:** Organizations can use this insight to reach out with their message to large number of people and also to listen actively to opinion leaders as ways to understand their customers needs and behaviors.
- **Public Health:** Awareness of networks can help identify the paths that certain diseases take to spread.

Network Topologies

- Two primary network topologies are

a) **Ring type**

b) **Hub Spoke topologies**

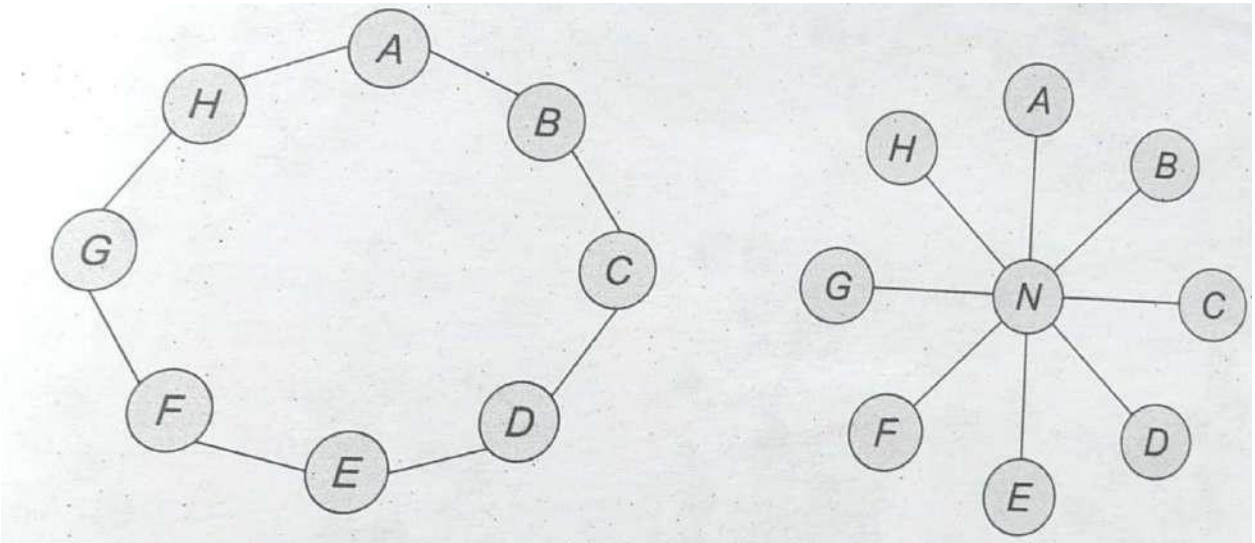


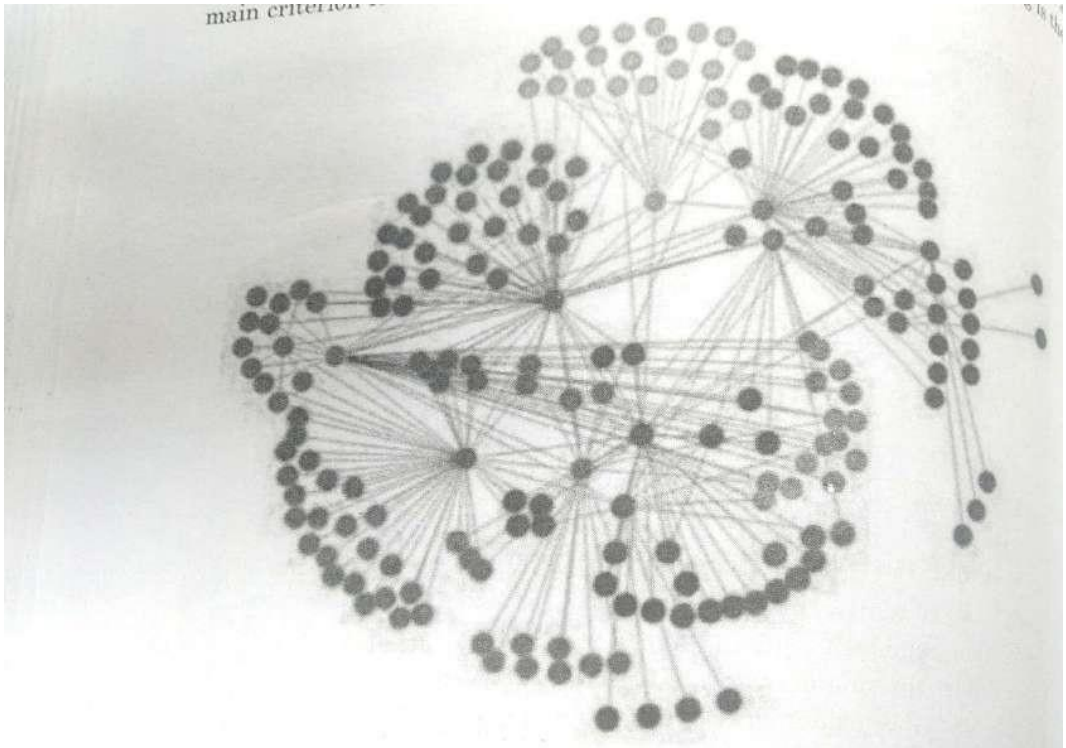
FIGURE 15.1 Network Topologies: Ring (left) and Hub-spoke (right)

- Ring network : Nodes typically connect to their adjacent nodes in the network.
- All nodes can be connected to each other.
- Connection can be dense or sparse.
- A dense network with more connections will allow many direct connections between pairs of nodes.

- In a sparse network one may need to traverse many connections to reach desired destinations.
- Peer to Peer email is an example of ring model.
- It is sparse as people connect directly with only a subset of people.

Hub Spoke Model

- This is a hierarchical network structure since the hub node is central to the network.
- The hub node is structurally more important as it is central to all communications between other peripheral nodes.
- Hub spoke network could predictably reach from any node to any other node through traversing exactly just two connections.
- Modern Airlines operate on this model.
- Finding sub networks



Source: GrapLab Inc.

FIGURE 15.2 A Network with Distinct Subnetworks

Consider the following simple network with 4 nodes (A, B, C, D) and 6 directed links between them as shown in Figure 15.3. Note that there is a bidirectional link. Here are the links

Node A links into B
Node B links into C
Node C links into D
Node D links into A
Node A links into C
Node B links into A.

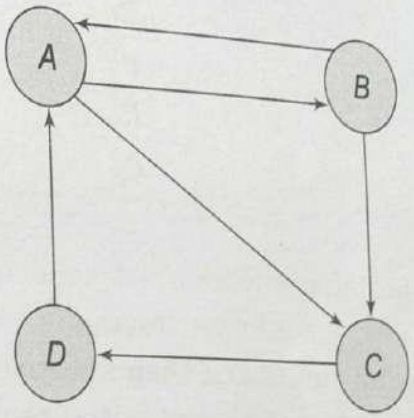


FIGURE 15.3

The goal is to find the relative importance, or rank, or every node in the network. This will help identify the most important node(s) in the network.

We begin by assigning the variables for influence (or rank) value for each node, as Ra, Rb, Rc, and Rd. The goal is to find the relative values of these variables.

There are two outbound links from node A to nodes B and C. Thus, both B and C receive half of node A's influence. Similarly, there are two outbound links from node B to nodes C and A, so both C and A receive half of node B's influence.

There is only outbound link from node D into node A. Thus, node A gets all the influence of node D. There is only outbound link from node C into node D and hence, node D gets all the influence of node C.

Node A gets all of the influence of node D and half the influence of node B.

Thus, $Ra = 0.5 * Rb + Rd$

Node B gets half the influence of node A.

Thus, $Rb = 0.5 \times Ra$

Node C gets half the influence of node A and half the influence of node B.

Thus, $Rc = 0.5 \times Ra + 0.5 \times Rb$

Node D gets all of the influence of node C and half the influence of node B.

Thus, $Rd = Rc$

We have 4 equations using 4 variables. These can be solved mathematically.

We can represent the coefficients of these 4 equations in a matrix form as shown in Dataset 15.1 given below.. This is the Influence Matrix. The zero values represent that the term is not represented in an equation.

Dataset 15.1

	Ra	*Rb*	*Rc*	*Rd*
Ra	0	0.50	0	1.00
Rb	0.50	0	0	0
Rc	0.50	0.50	0	0
Rd	0	0	1.00	0

For simplification, let us also state that all the rank values add up to 1. Thus, each node has a fraction as the rank value. Let us start with an initial set of rank values and then iteratively compute new rank values till they stabilize. One can start with any initial rank values, such as $1/n$ or ¼ for each of the nodes.

Variable	Initial Value
Ra	0.250
Rb	0.250
Rc	0.250
Rd	0.250

Computing the revised values using the equations stated earlier, we get a revised set of values shown as Iteration1. (This can be computed easily by creating formulae using the influence matrix in a spreadsheet such as Excel.)

Variable	Initial Value	Iteration1
Ra	0.250	0.375
Rb	0.250	0.125
Rc	0.250	0.250
Rd	0.250	0.250

Using the rank values from Iteration1 as the new starting values, we can compute new values for these variables, shown as Iteration2. Rank values will continue to change.

Variable	Initial Value	Iteration1	Iteration2
Ra	0.250	0.375	0.3125
Rb	0.250	0.125	0.1875
Rc	0.250	0.250	0.250
Rd	0.250	0.250	0.250

Working from values of Iteration2 and so, we can do a few more iterations till the values stabilize. Dataset 15.2 shows the final values after the 8th iteration.

Dataset 15.2

Variable	Initial Value	Iteration1	Iteration2	...	Iteration8
Ra	0.250	0.375	0.313	...	0.333
Rb	0.250	0.125	0.188	...	0.167
Rc	0.250	0.250	0.250	...	0.250
Rd	0.250	0.250	0.250	...	0.250

The final rank shows that rank of node *A* is the highest at 0.333. Thus, the most important node is *A*. The lowest rank is 0.167 of *Rb*. Thus, *B* is the least important node. Nodes *C* and *D* are in the middle. In this case, their ranks did not change at all.

The relative scores of the nodes in this network would have been the same irrespective of the initial values chosen for the computations. It may take longer or shorter number of iterations for the results to stabilize for different sets of initial values.

Pager Rank

- Is a particular application of SNA above to compute the relative importance of website in the overall WWW.
- Every web page is a node in a social network and all the hyperlinks from that page become directed links to other webpages.

- An iterative computational technique is applied to compute a relative importance to each page.
- That score is called PageeRank
- PageRank is used by Google for ordering the display of websites in response to search queries.
- Teleporting factor : Whether a website actually exist. Influence matrix is multiplied by a weighting factor called beta with a value of .85 or 85% .
- Remaining .15 or 15% is given teleportation.
- Teleportation matrix each cell is given a rank of 1/n where n is the number of nodes in a web.
- Two matrices are added to compute the final influence matrix.
- This matrix can be used to iteratively compute the PageRank of all the nodes.

Practical Considerations

- **Network Size:** Most SNA research is done using small network. Collecting data about large networks can be very challenging.
- Because the number of links is the order to square of the number of nodes. Thus in a network of 1000 nodes there are potentially 1 million possible pairs of links.
- **Gathering Data:** Electronic Communications records (emails, chats etc) can be harnessed to gather social network data more easily.
- Data on nature and quality of relationships need to be collected using survey documents.
- Capturing and cleansing and organizing the data can take a lot of time and effort , just like typical data analytics project.
- **Computation and Visualization:** Modeling large networks can be computationally challenging and visualizing them also would require special skills.
- Big data analytical tools may be needed to compute large networks.
- **Dynamic Networks:** Relationships between nodes in a social network can be fluid.
- They can change in strength and functional nature.

Table: Social Network Analysis vs Traditional Data Analytics

Dimensions	Social Network Analysis	Traditional Data Mining
Nature of learning	Unsupervised learning	Supervised and unsupervised learning
Analysis of goals	Hub nodes, important nodes, and subnetworks.	Key decision rules, cluster centroids
Dataset structure	A graph of nodes and (directed) links	Rectangular data of variables and instances
Analysis techniques	Visualization with statistics iterative graphical computation	Machine learning statistics
Quality Measurement	Usefulness is key criterion	Predictive accuracy for classification techniques