NAVODAYA INSTITUTE OF TECHNOLOGY, RAICHUR

DEPARMENT OF COMPUTER SCIENCE & ENGINEERING

Module - 5

Spark architecture features, software stack components and their functions. Section 5.3 describes steps in data analysis with Spark and using Spark with advanced Python features. Section 5.4 describes methods of downloading Spark, programming with RDDs, Spark shell and developing and testing Spark codes, and applications of MLib. Section 5.5 describes ETL processes using the built-in functions and operators, and ETL pipelines. Section 5.6 describes data analytics, data reporting and data visualization.

5.21 SPARK

~pa1nk arcltrlitec~u ral featmes, soflrt.•.r.ue stack components and tliileir 1i"liln otiom;

Apache® Spark™ is a *fast and general compute engine.* Apache® Spark™ powers the analytics applications up to 100 times faster. It supports *HDFS compatible data.* Spark has a simple and *expressive programming model.*

Expressive program model implements a *number* of mathematical and logic operations with smaller and easier written codes which a compiler as well as programmer can understand easily. The model, therefore, gives programming ease for a wide range of applications. Applications of expressive codes are in analytics, Extract Transform Load (ETL), Machine Learning (ML), stream processing and graph computations.

Spark runs on both Windows and UNIX-likesystems, such as Linux and Mac OS.Java is essential for running Spark applications. Executing a spark application on a computer system therefore requires setting a JDK path using JAVA_HOME environment variable or system variable, PATH. Spark 2.3.1 runs onJava 8+, Python 2.7+/3.4+ and R 3.1+, and Scala 2.11.x.The multiple languages, Python and Scala shells provide *great ease in programming for complex analytics, machine learning* and other solutions.

Following subsections describe Spark and introduce data analysis using Spark.

5.2.1 Introductionto Big Data Tool-Spark

Figure 5.1 shows the main components in the Apache Software Foundation's Spark framework, which includes data storage, APis and resources management bonded with functions in Spark core.

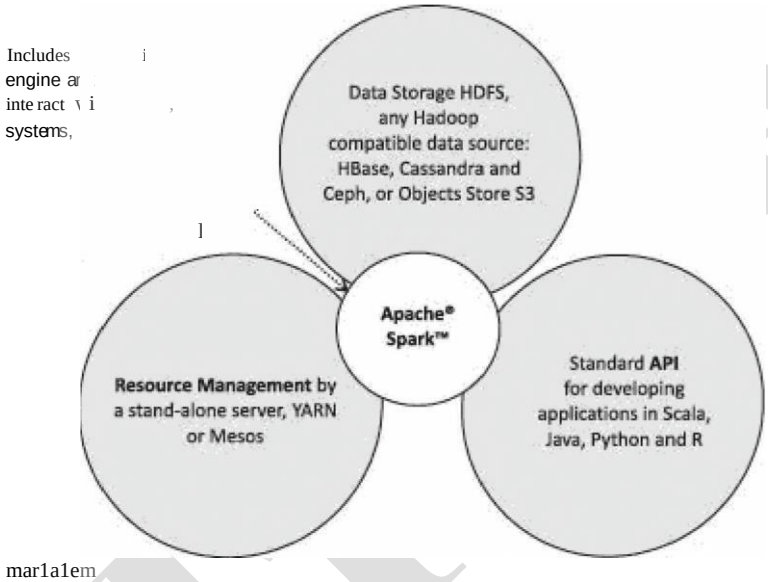


Figure 5.1 Main components of the Spark architecture

Main components of the Spark architecture are:

1. Spark HDFS file system for data storage: Storage is at an HDFS or Hadoop compatible data source (such as HDFS, HBase,Cassandra, Ceph), or at the Objects Store 53
2. Spark standard API enables the creation of applications using Scala,Java, Python and R
3. Spark resource management can be at a stand-alone server or it can be on a distributed computing framework, such as YARN or Mesos.

Ceph refers to an open source, scalable object, block, file storing unified system. Ceph provides for dynamic replication and redistribution. Ceph provides HDFS compatible mechanisms for Big data in petabytes or exabytes. An application uniquely accesses the object, block and file stores in a system using Ceph. Ceph is highly reliable.

Apache Mesos is an open source project developed at University of Berkeley, and now passed to Apache. It manages computing clusters. Its implementation is in C++. Apache released a stable version 1.3.0 of Mesos in June 2017. Apache Mesos enables fine-grained sharing of CPU, RAM, IOs and other across frameworks. Mesos offers them resource. Each resource contains a list of agents. Each agent has the Hadoop and MapReduceexecuters.

53 (Simple Storage Service) refers to an Amazon web service on the cloud named 53 which provides Object Stores for data (Section 3.3.4).

5.2.1.1 Features of Spark

Figure 5.2 shows the main features of Spark.

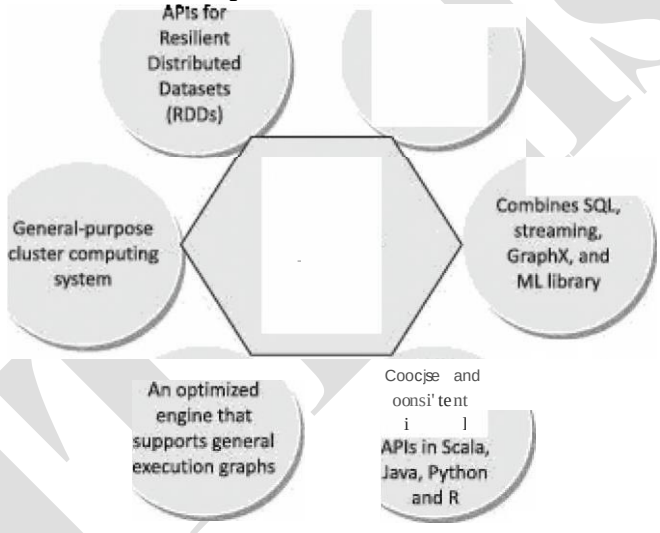


Figure 5.2 Main features of Spark

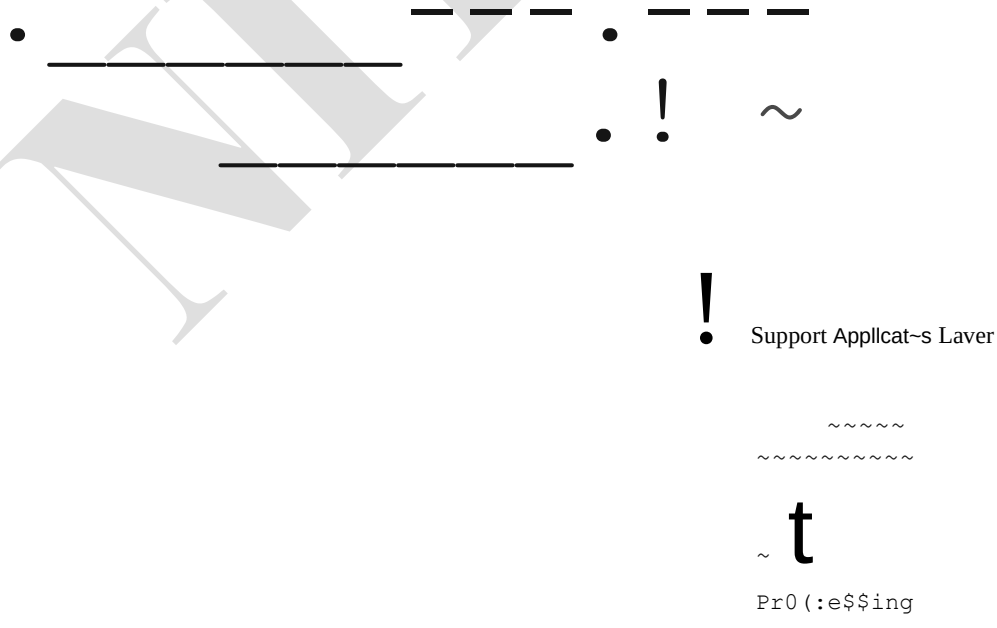
The features of Spark:

1. Spark provisions for creating applications that use the complex data. In-memory Apache Spark computing engine enables up to 100 times performance with respect to Hadoop.
2. Execution engine uses both in-memory and on-disk computing. Intermediate results save in-memory and spill over to disk.
3. Data uploading from an Object Store for immediate use as a Spark object instance. Spark service interface sets up the Object Store.
4. Provides high performance when an application accesses memory cache from the disk.

5. Contains API to define Resilient Distributed Datasets (RDDs). ROD is a programming abstraction. ROD is the core concept in Spark framework. ROD represents a collection of Object Stores distributed across many compute nodes for parallel processing. Spark stores data in ROD on different partitions. A table has partitions into columns or rows. Similarly, an ROD can also be considered as a table in a database that can hold any type of data. RDDs are also fault tolerant.
6. Processes any kind of data, namely structured, semi-structured or unstructured data arriving from various sources.
7. Supports many new functions in addition to Map and Reduce functions.
8. Optimizes data processing performance by slowing the evaluation of Big Data queries.
9. Provides concise and consistent APIs in Scala, Java and Python. Spark codes are in Scala and run on JVM environment.
10. Supports Scala,Java, Python, Clojure and R languages.
11. Provides powerful tool to analyze data interactively using shell which is available in either Scala (which runs on the Java VM and is thus a good way to use existing Java libraries) or Python. The tool also provides for learning the usages of APL

5.2.1.2 Spark Software Stack

Figure 5.3 shows a five-layer architecture for running applications when using Spark stack.



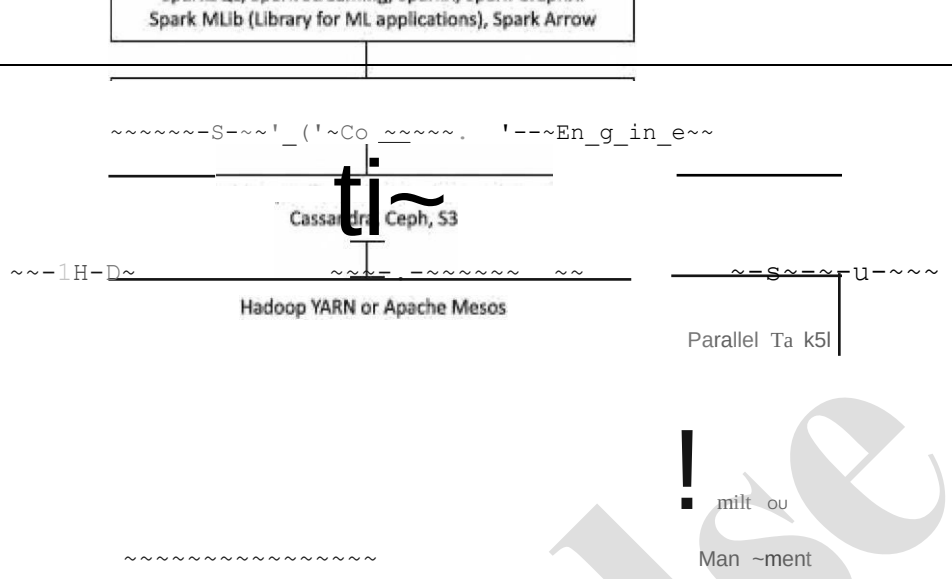


Figure 5.3 Five-layer architecture for running applications using Spark stack

The main components of Spark stack are SQL, Streaming, R, Graphx, MLib and Arrow at the applications support layer. Spark core is the processing engine. Data Store provides the data to the processing engine. Hadoop, YARN or Mesos facilitates the parallel running of the tasks and the management and scheduling of the resources.

Spark Stack

Spark stack imbibes generality to Spark. Grouping of the following forms Spark stack:

Spark SQL for the structured data. The SQL runs the queries on Spark data in the traditional business analytics and visualization applications. Spark SQL enables Spark datasets to use JDBC or ODBC APL HQL queries also run in Spark SQL. Runs UDFs for inline SQL, distributed DataFrames, Parquet, Hive and Cassandra Data Stores.

Spark Streaming is for processing real-time streaming data. Processing is based on micro-batches style of computing and processing. Streaming uses the DStream which is basically a series of RDDs, to process the real-time data.

SparkR is an R package used as light-weight front end for Apache Spark from R. Spark API uses SparkR through the RDD class. A user can interactively run the jobs from the R shell on a cluster. An RDD API is in the distributed lists in R.

Spark MLibis Spark's scalable machine learning library. It consists of common learning algorithms and utilities. MLib includes classification, regression, clustering, collaborative filtering, dimensionality reduction and optimization primitives. MLib applies in recommendation systems, clustering and classification using Spark.

Spark Graphx is an API for graphs. Graphx extends the Spark RDD by introducing the Resilient Distributed Property. GraphX computations use fundamental operators (e.g., subgraph, joinVertices and aggregateMessages). GraphX uses a collection of graph algorithms for programming. Graph analytics tasks are created with ease using GraphX.

Spark Arrow for columnar in-memory analytics and enabling usages of vectorised UDFs (VUDFs). The Arrow enables high performance Python UDFs for SerDe and data pipelines.

Self-Assessment Exercise linked to LO 5.1

1. How does Spark process as a fast and general compute engine? What does expressive programming model mean?
2. List the main components of Spark stack and the functions of each.
3. List the advanced provisions in Spark SQL compared to HiveQL.
4. List the main features of Spark?

5.31 INTRODUCTION TO DATA ANALYSIS WITH SPARK

"Analysis of data is a process of inspecting, cleaning, transforming and modeling data with the goal of discovering useful information, suggesting conclusions and supporting decision-making." *(Wikipedia)*

LO 5.2

Steps i 111 olata, aliTlalrysts with Spa~k, usl ngi Spar.1k w.lth tP)i'ih 0111 3rnl311ilcedl fenu res, 'U DFs, vector,ized <IJl[):Fs, group vectori1zed ILJ [!!1Fs and Pvtlmon, I ibr,a ri1es for t:h *e* ana11ysis

For examples, consider a car company. Assume that company sells five models of cars. Each model is available in 16 different colours and shades. Sales are processed through a large number of showrooms, all over the country. An analyses of annual sales and annual profits model-wise, colour-wise, region• wise and showroom-wise help the company in discovering useful information and making suggestive conclusions. The company does predictive analytics from the results of the analyses and decides the future strategies for manufacturing, sales and out-reaches to customers on the basis of the results of the analytics.

Figure 5.4 shows the steps between acquisition of data from different sources, applications of the analyzed data, and application support by Spark for the analyses.

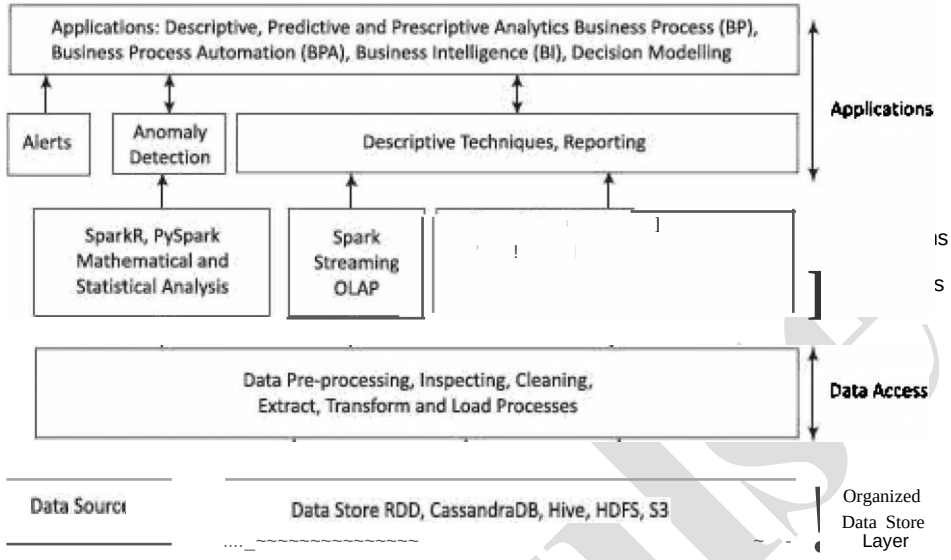


Figure 5.4 Steps between acquisition of data from different sources and its applications

Following are the steps for analyzing the data:

1. Data Storage: Store of data from the multiple sources after acquisition. The Big Data storage may be in HDFS compatible files, Cassandra, Hive, HDFS or S3.
2. Data pre-processing: This step requires:

   (a) dropping out of range, inconsistent and outlier values,

   (b) filtering unreliable, irrelevant and redundant information,

   (c) data cleaning, editing, reduction and/or wrangling,

   (d) data-validation, transformation or transcoding.

3. Extract, transform and Load (ETL))for the analysis
4. Mathematical and statistical analysis of the data obtained after querying relevant data needing the analysis, or OLAP,
5. Applications of analyzed data, for example, descriptive, predictive and prescriptive analytics, business processes (BPs), business process automation (BPA), business intelligence (BI), decision modelling and knowledge discovery.

5.3.1 Spark SQL

Spark SQL is a component of Spark Big Data Stack. Spark SQL components are DataFrames (SchemaRDDs)S, QLContextandJDBCserver. Spark SQL at Spark does the following:

1. Runs SQL like scripts for query processing, using *catalyst optimizer* and *tungsten execution engine*
2. Processes structured data
3. Provides flexible APis for support for many types of data sources
4. ETL operations by creating ETL pipeline on the data from different file-formats, such asJSON,Parquet, Hive, Cassandra and then run ad-hoc querying.

Spark SQL has the following features for analysis:

1. SparkR, PySpark, Python, Java and other language support for coding for data analysis.
2. Provisioning of JDBC and ODBC APis: Applications in Java and Microsoft programs (such as Excel) need to connect to databases using JDBC (Object Database Connectivity) and ODBC (Object Database Connectivity). Spark APis enable that connectivity,
3. Spark SQL enables users to extract their data from different formats, such as Hive, *JSON* and Parquet, and then transform that into required formats for ad hoc querying.

[Ad hoc query is a query 'just for this purpose' or query 'on the fly.' For example, var newToyQuery = "SELECT * FROM table WHERE id = " + toy_puzzleId. The result will be different each time this code executes, depending on the value of toy_puzzleId.

4. Spark SQL processing support inclusion of Hive. Hive support enables the use of Hive tables, database and data warehouse, UDFs and SerDe (serialization and deserialization).
5. Spark SQL supports HiveQL and Cassandra CQL for query processing.
6. Spark Streaming for support to OLTP and structured streaming.

Figure 5.5 shows the connectivity of applications with Spark SQL which connects to Object Stores in different formats.

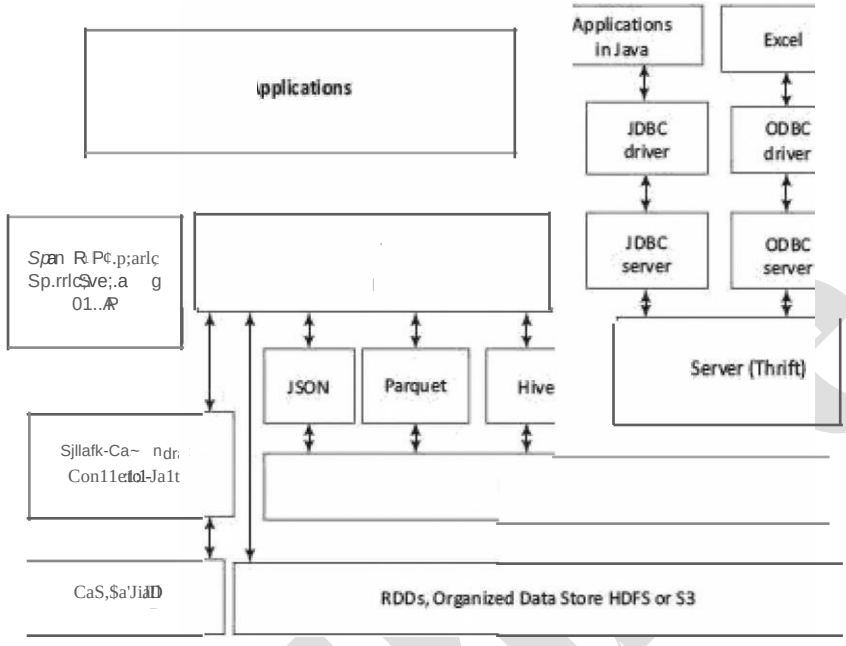


Figure 5.5 Connectivity between the applications and Spark SQL

JDBC Server An application reads the data tables in RDBMS using a JDBC client 0DBC API at the application). Many applications in Java connect to databases using JDBC driver and server. Spark SQL API provides JDBC connectivity. Command for using JDBC server is as follows:

```
./sbin/start-thriftserver.sh  -- master sparkMaster
```

Hive Server (Thrift) enables a remote Hive client or JDBC driver to send a request to Hive and the server sends response to that (Section 4.4.1). The requests can be in Scala, Java, PythonorR.

JSON, Hive, Parquet Objects Section 3.3.2 explained JSON object data formats and files. Section 4.4 explained Hive, HiveQL database and QL commands for data definition of databases, tables, columns, partitions and views, and their querying.

HDFS is highly reliable for very long running queries. However, IO operations are slow. Columnar storage is a solution for faster IOs. Columnar storage stores the data portion, presently required for the IOs. Load-only columns access during processing. Also, a columnar object Data Store can be compressed or encoded according to the data type. Also, executions of different columns or column partitions can be in parallel at the data nodes.

Section 3.3.3.3 and Section 3.3.3.4 described record columnar (RC) file, and optimized row columnar (ORC) file formats respectively. Hive RC file records store in columns and can be partitioned into row groups. An ORC file consists of row-groups row data called stripes. ORC enables concurrent reads of the same file using separate RecordReaders.1 Metadata stored using protocol buffers for addition and removal of fields.

Parquet is a nested hierarchical columnar storage concept. Apache Parquet file is a columnar storage file (Section 3.3.3.5).The file uses an HDFS block. The block saves the file for running big long queries on Big Data. Each file compulsorily consists of metadata, though a file need not consist of data. An application retrieves the columnar data quickly from Parquet files.

Apache Parquet three projects specify the usages of files for query processing or applications. The projects are (i) *parquet-format* for specifying formats and Thrift definitions of metadata, (ii) *parquet-mr* for implementing the sub-modules in the core components for reading and writing a nested, column-oriented data stream, and (iii) *parquet-compatibility* for compatibly for read-write in multiple languages.

Spark DataFrame (SchemaRDD) A DataFrame is a distributed collection of data organized into named columns. DataFrame can be used for transformation using filter, join, or groupby aggregation functions.

Example 5.8 in Section 5.4.2 will explain schema creation for DataFrames and usage of RDDs. Earlier, DataFrame in Spark was called SchemaRDD. Section 5.4.2 describes SchemaRDDand creation of RDDs from row objects. An RDD method converts Spark DataFrames to RDDs. Each RDD consists of a number of row objects.

Creating Spark DataFrame (SchemaRDD) from Parquet andJSON Objects DataFrames can be created from different data sources. Examples of data sources are *JSON* datasets, Hive tables, Parquet row groups, structured data files, external databases and existing RDDs. Section 10.4 will describe Hive and PySpark programs using functions, Merge and Join in Dataframes of large datasets.

The following example explains the creation and usages of Dataframes from the Parquet and *JSON* objects:

EXAMPLE 5.1

SOLUTION

(i) The following statement creates sqlContext from Spark Context sc:

```
SqlContext                new
                          sqlContext
```

Assume a table toyPuzzleTypeCostTblwith four columns. Figure 5.6 shows the sample

table toyPuzzleTypeCostTbl. The columns are puzzle type, puzzle code, number of puzzle pieces and puzzle cost. DataFramel named toyPuzzleTypeCodes consists of columns 1 and 2. DataFrame2 named toyPuzzleCodesCostconsists of Columns 2 and 4. The table consists of multiple rows in each row group. The dashed lines point to the columns in the two data frames. toyPumeTypeCmtlbl

Figure 5.6 Sample table toyPuzzleTypeCostTblrows, row groups and DataFrames

(i) Create a sqlContext from a given SparkContext 'sc'.

(ii) Create a four columns DataFrame using the Parquet file named "toyPuzzleTypeCostTbl".

(iii) How will a DataFrame create using a JSON file format file "toyPuzzleTypeCostTbl"? Create two DataFrames using Java and SqlContext, one with columns for puzzle type and puzzle code, and the other for puzzle code and cost.

(iv) How will two DataFramesjoin using puzzle code as a join key to create a DataFrame of three columns, 1, 2 and 4?

```
org.apache.spark.sql.SQLContext(sc)
```

(ii) The following statement creates a DataFrame named toyTypeCost:

```
DataFrame                    toyTypeCost
sqlContext.parquetFile("toyPuzzleTypeCostTbl")
```

[DataFrame created will have four columns.]

(iii) DataFrame creates using Load()method at the sqlContext.

```
# To display the contents of Table: "toyPuzzleTypeCostTbl"

spark.sql ("SELECT*  FROM toyPuzzleTypeCostTbl  ")

# To create DataFrarne toyPuzzleTypeCostTbl

DataFrarne toyPuzzleTypeCostTbl          sqlContext.Load
 ("toyPuzzleTypeCostTbl", "json")
```

The following statements create two DataFrames using DataFrame toyPuzzleTypeCostTbl.One frame of two columns, 1 and 2, puzzle type and puzzle code:

```
DataFrarne    toyTypeCodes          toyPuzzleTypeCostTbl.
select(toyPuzzleTypeCostTbl          [  'puzzleType' J,
toyPuzzleTypeCostTbl  ['puzzleCode'J)
```

Second frame of two columns, 2 and 4, puzzle code and puzzle cost

```
DataFrarne                toyCodesCost
toyPuzzleTypeCostTbl.select(toyPuzzleTypeCostTbl
['puzzleCode'], toyPuzzleTypeCostTbl  ['puzzleCost'J)
```

(iv) DataFramel (toyTypeCodes) has two columns, 1 and 2, as *puzzleType* and *puzzleCode*, respectively. The frame joins dataFrame2 (toyCodesCost)column 4 as *puzzleCost* and resultant is a joined new-dataFrame *toyTypeCodesCost* consisting of columns 1, 2 and 4. Join key is *puzzleCode.* Following statements use join() method and joining key.

Format of the statement is

dataFrame. join (dataframel, dataframe2.col *("JoinKey")* dataframeNew ("]oinKey")

and the statement is

```
dataFrarne.join(toyTypeCodes,
toyCodesCost.col("puzzleCode").
equalTo (toyTypeCodesCost ("puzzleCode")))
```

Using HiveQL for Spark SQLSpark SQL programming provides two contexts, SQLContext and HiveContext. While using HiveContext, then, commands access the Hive Server only and use HiveQL commands. SQLContextis a subset of Spark SQL. SQLContextdoes not need the HiveServer (Thrift). Therefore, when needing access to the HiveServer, specify the HiveContext. HiveQL is recommended for Spark SQL. Many resources are available in Hive readily and can directly be used.

Use of Aggregation and Statistical Functions Aggregation functions can be used for analysis. Hive consists of count (*), count (expr); sum (col), sum (DISTINCT col), avg (col), avg (DISTINCT col), min (col) and DOUBLE max(col) (Table 4.10). The statistical functions stdev(), sampleStdev(), variance, sampleVariance() can be used for analysis with DataFrames in input.

Consider the example below to find sum of the sales of the Jaguar Land Rover model and trace the showroom that recorded the best sales.

EXAMPLE 5.2

Learning Outcomes

Recall Practice Exercise 3.11. Consider the annual car sales data in the following format:

			115	38
ca	rShow room	sc umul ative	Year ly	Sa les
				88

Hexa Sale (HDS)	*Zest* Sales (ZDS)	*Nexon* Sale (NDS)

Car ShowroomID (csID)

") 0

]212J7	49	34	164
nnn	40	14]	123 37

(i) Find the Jagaur Land Rover annual sale figure over all showrooms.

(ii) Find the showroom ID and Land Rover sales figure for the showroom giving *maximum Jaguar Land Rover sales.*

SOLUTION

(i) The following query returns the sum of the Jagaur Land Rover annual sale in column 3 of the table:

```
SELECT sum (JLRDS)FROM CarShowroomsCumulativeYearlySales
```

(ii) The following nested query statement returns the csID and maximum annual sale value for Land Rover:

```
  SELECT csID, saleJL (max (map (csID, ID,  JLRDS, saleJL)))
FROM CarShowroomsCumulativeYearlySales
```

5.3.2 Using Python Advanced Features with Spark SQL

Python is a general purpose, interpreted, interactive, object oriented and high level programming language. Python defines the basic data types, containers, lists, dictionaries, sets, tuples, functions and classes. Python Standard Library is very extensive. The libraries for regular expressions, documentation generation, unit testing, web browsers, threading, databases, CGI, email, image manipulation and a lot of other functionalities are available in Python.

Python programming is a strong combination of performance and features in the same bundle of codes. Spark SQL binds with Python easily. Python has the expressive program statements. Spark SQL features together with Python help a programmer to build challenging applications for Big Data. The following example explains the use of PySpark, Python along with the Spark SQL:

EXAMPLE 5.3

(i) How is HiveContext and Spark SQL used?

(ii) How does PySpark use a row object?

(iii) Assume a JSON file, *toyTypeProductTbl* of row objects. Assume the use of a row object *toyPuzzleProduct* for query (Table 4.13). How do a file load and query sent to the file?

ProdndCate ry	Produ ld	Prodn Name
Row Objectl	To _A:irplruie	Lost Temple
Row Object:'.	Toy_Airplill.!lle	
Row bje t3		

SOLUTION

(i) Import Spark SQL using the following statement:

```
Import Spark SQL
```

(ii) Then use the following command for importing a row object, toyPuzzleTypeCost:

```
# Now use the variable named hiveCtx using Hive Context
from
# sc statement using statement
hiveCtx = HiveContext (sc)
from pyspark.sql import HiveContext, toyPuzzleTypeCost
```

(iii) A file loads using hiveCtx and input loads at the file

```
input=  hiveCtx.jsonFile(toyTypeProductTbl)

input.registerTempTable  ("toyPuzzleProduct")
```

The queries raised for finding Product_ID_Name using SELECT command.

```
Product_ ID_Name      hiveCtx. sql  ( "SELECT  ID, name  FROM
toyPuzzleProduct ORDER BY ProductCategory")
```

5.3.2.1 Python Libraries for Analysis

NumPy and SciPyare open source downloadable libraries for numerical (Num) analysis and scientific (Sci) computations in Python (Py). Python has open source library packages, NumPy, SciPy, Scikit-learn, Pandas and StatsModel, which are widely used for data analysis. Python library, matplotlib functions plot the mathematical functions.

Spark added a Python API support for UDFs. The functions take one row at a time. That requires overhead (additional codes) for SerDe.Earlier data pipelines first defined the UDFs in Java or Scala, and then invoked them from Python. Spark 2.3 provisions for vectorized UDFs (VUDFs) and Apache Arrow facilitates VUDFs, which enables high performance Python UDFs for SerDe and data pipelines.

NumPy NumPy includes (i) N-dimensional array object, array and vector mathematics; (ii) linear algebraic functions, Fourier transform and random number functions; (iii) sophisticated (broadcasting) functions; and (iv) tools for integrating with C/C++ and Fortran codes.

NumPy provides multi-dimensional efficient containers of generic data and definitions of arbitrary data types. NumPy integrates easily with a wide variety of databases. NumPy provides import, export (load/save) files, creation of arrays, inspection of properties, copying, sorting and reshaping, addition and removal of elements in the arrays, indexing, sub-setting and slicing of the arrays, scalar and vector mathematics (such as +, - , x, +, power, sqr, sin, log, ceil - round up to nearest int, floor - round down up to the nearest int, round-round to nearest integer). NumPy also provides statistical functions.

Table 5.1 gives the examples of NumPy functions for data analysis problems.

Table 5.1 Examples of NumPy functions for data analysis problems

Function	Description	Function	Description

np.loadtxt('file.txt')	Loads a text file	np.mean(arr, axis= o)	Returns mean along a specific axis
np.sum(); Returns the sum and np.genfromtxt('file .csv', Loads a csv file with comma as the delimiter=',')	delimiter between records	np.minl),	minimum of the array
np.savetxt('file.txt,' arr, delimiter='')	Saves a text file which is an array of strings separated by a space each.	np.max(arr, axis= o)	Returns the maximum along a specific axis
np.genfromtxt('file .csv', arr, delimiter=',')	Saves a CSV file which is an array of strings separated by a comma each.	np.var(arr)	Returns the variance of array
arr.sort()	Sorts an array	np.std(arr, axis= o)	Returns the standard deviation of a specific axis
np.add (arrl, arr2)	Performs a vector addition of array 1 and array 2	np.corr()	Returns the correlation coefficient

SciPy SciPy adds on top of NumPy. It includes MATLAB files and special functions, such as routines for numerical integration and optimization. SciPy defines some useful functions for computing distances between a set of points.

SciPy includes (i) interactions with NumPy, (ii) creation of dense and open mesh grids, (iii) shape manipulation functions, (iv) polynomial and vectoring functions, (v) real and imaginary functions, and casting an object to a data type, and (vi) matrix creation and matrices routines and usages of spark matrices.

Table 5.2 gives few examples of SciPyfunctions for scientific computational problems.

Table 5.2 Examples of SciPyfunctions for data analysis problems

Function	Description	Function	Description
np.c jb, c]	Create Stacked column-wise array	np.cast ['f] (np.pi)	Casts an object into a data ype
		from numpy	

b.flatten()	Flattens the array	import poly ID p = polyID ([2, 3, 4])	Creates a polynomial object p
np.vsplit (c, 2) and np.hsplit (d, 2)	Functions for vertically splitting and horizontally splitting the array at the end of the second index	A.I, A.T, A.H	Inverses, transposes, and conjugate transposes the matrix, A
linalg.det(A)	Returns the determinate of A	np.select ([c<4], [c*2])	Returns values from a list of arrays depending on the conditions
np.img(c)	Returns the imaginary part of the array elements	A np.matrix ([3, 4], [5, 6])	Creates a matrix

Panda Panda derives its name from usages of a data structure called Panel. The first three characters *Pan* in Panda stand for the term 'panel'. The next two characters *da* in Panda stand for data. The Panda package considers three data structures: Series, DataFrame and Panel.

DataFrame is a container for Series. Panel is a container for DataFrame objects. The Panel objects can be inserted or removed similar to as in a dictionary. DataFrame may be a DataFrame of statistical or observed datasets. The data need not be labeled. This means datasets, objects and DataFrames can be placed into a Panda data structure without labels.

Panel is a container for three-dimensional data. Panel is a widely used term in econometrics. Three axes describe operations involving panel data. For example, panel data in econometric analysis. Items can be considered as along axis O of an inside DataFrame (set of columns). Index (rows) of each of the DataFrames correspond to axis 1 (major axis). Columns of each of the DataFrames correspond to axis 2 (minor axis). Panel 4D consists of labels as O - axis, items - axis 1, major axis - axis 2 and minor axis - axis 3. Figure 5.7 shows the main features of Pandas package.

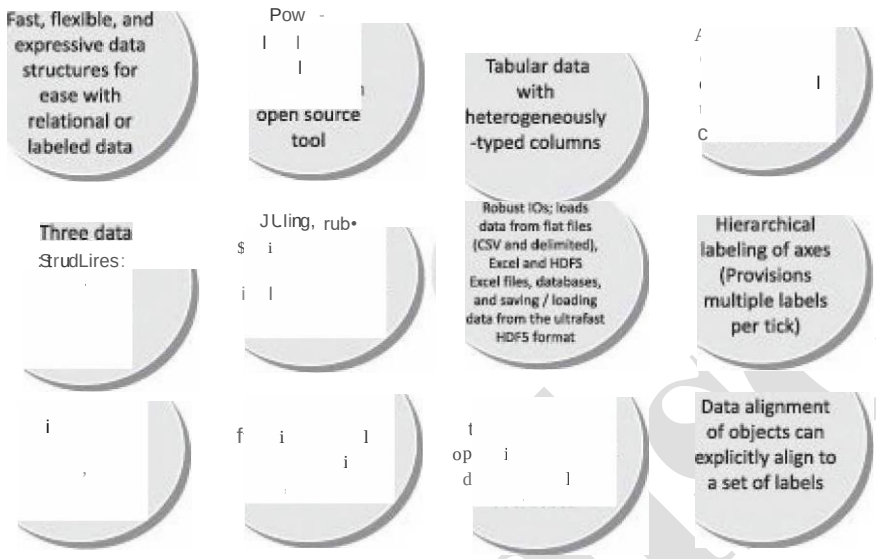


Figure 5. 7 Main features of Panda for data analysis

Pandas package includes the following provisions:

1. Database style Dataframes merge, join, and concatenation of objects
2. RPy interface for R functions plus additional functions
3. Panda ecosystem has statistics, machine learning, integrated development environment (IDE), API and several out of core features
4. SQL like features: SELECT, WHERE, GROUPBY,JOIN, UNION, UPDATE and DELETE
5. GroupByfeature of split-apply-combine with the steps as: (i) an object such as table, file or document splits into groups, (ii) iterate through the groups and select a group for aggregation, transformation and/or filtration. The instance method can be dispatched and applied in a manner similar to the aggregation/transformation function
6. Size mutability, which means that columns can be inserted and deleted from DataFrame and higher dimensional objects
7. Slicing and dicing a collection of DataFrame objects. The names of axes can be somewhat arbitrary in a Panel. (Arbitrary means the axes need not be named al, a2, ... , an, and can be year, car model, sales, ...). If slicing function slices the first dimension, the lower dimension objects are obtained.

5.3.2.2 User-Defined Functions (UDFs)

The functions take one row at a time. This requires overhead for SerDe. Data exchanges take place between Python and JVM. Earlier the data pipeline (between data and application) defined the UDFs in Java or Scala, and then invoked them from Python while using Python libraries for analysis or other application. SparkSQLUDFs enable registering of themselves in Python,Java and Scala.

The SQL calls the UDFs. This is a very popular way to expose advanced functionality to SQL users. User codes call the registered UDFs into the SQL statements without writing the detailed codes. The following example demonstrates how a UDF is created in PySpark.

EXAMPLE 5.4

Recapitulate Example 5.1 table, toyPuzzleTypeCostTbl. Create a UDF, udfCostPlus () in pandas. The table column puzzleCost creates using jigsaw_puzzle_info.txt from an RDD. Write a UDF which increases the costs in the column, puzzle_cost_uso by 10%.(The UDF takes one row at a time as input.)

SOLUTION

Following are the Python statements

```
from pyspark.sql.functions  import udf
[Useudf to define a row at a time udf.]
@udfCostPlus('float')
[Input and output costs are two values both for a single float variable, v.]
def plusTenPercent(v):
return v + 0.1 xv; df.withColumn('v4', puzzle
cost_USD (df.v))
```

[Data Frame df has v4 as puzzleCost in the fourth column.]

Dataset at Example 5.2 consists of car sales data. Column 1 represents car showroom ID. The ID key is present in all date fields on which the sales were recorded during the year. Corresponding to an ID, column 3 has the Jaguar Land Rover sales figures in more than 300 rows for more than 300 dates. Sales figures of four other models are in columns 4, 5, 6 and 7.

The product sales analysis is widely performed in many businesses. Writing a UDF for analysis of sales once and using it for different products whenever and wherever desired reduces coding efforts. This also integrates new functions (UDFs) in higher level language with their lower level language implementations. Also, writing UDFs are helpful when built-in functionalities in a currently used tool needs additional functionalities.

A UDF using aggregation function max() calculates the Land Rover sales and traces the showroom giving *maximum Jaguar Land Rover sales.* The UDF is of great help to perform similar analyses on different models of the car.

Java class can be created user defined method (UDF) productSalesAnalysis (). Python scripts can also create the UDF to find that sales point ID and total yearly sale for that sales point from which the total is highest for a product. The UDF will be reusable not only for car sales analysis but also for analysis of sales of many companies, such as ACVM or Toy Company.

Python provides a register function: hiveContext {sc) . registerFunction {). The command can be hiveContext {sc). registerFunction {"csIDnl", int: bestYearlySalesModell, LongType (), ("csIDn2", int: bestYearlySalesModel2, LongType {), ...) . cslDnl is car-showroom IDl,
bestYearlySalesModell is best yearly sale of model 1.

5.3.2.3 Vectorized User Defined Functions (VUDFs)

Python UDFs express data in detail. Therefore, Python UDFs, block-level UDFs with block• level arguments and return types, conversions or transformations are widely used in ETL or ML applications. Spark Arrow facilates columnar in-memory analytics, which results in high performance of Python UDFs, SerDe and data pipelines.

VUDFs use series data structure (meaning one-dimensional array or tuples). Spark 2.3 (2018) provisions for using vectorized UDFs (VUDFs). Apache Arrow 0.8.0 (release date December 18, 2017) facilitates usages of VUDFs. Pandas UDF, *pandas_ UDF* uses the function to create a VUDF with (i) pandas.Series as input to the UDF, (ii) pandas.Series as output from the UDF, (iii) no grouping using GroupBy, (iv) output size same as input, and (v) returns the same data types as specified type in return pandas.Series.

The following example explains the use of VUDF.

EXAMPLE 5.5

Recapitulate Example 5.1 toyPuz z leTypeCos t Tbl. Create a vectorized UDF (VUDF). First define a pandas_UDFCostPlusfor increasing cost puzzle_cost_USD of toys in puzzle_Costs RDD created from jigsaw_puzzle_info.txt.,

SOLUTION

Following are the Python statements from pyspark.sql.functions import pandas_udf:

```
from pyspark.sql.functions  import pandas_udf
```

[Usepandas_udf to define a vectorized udf.]

```
@pandas_udfCostPlus  ('float')
```

[Input/ output are both a pandas.Series of elements with data type *float.]*

```
def vectorized_plusTenPercent  (v):
return v4 + 0.1 df.withColumn('v4', vectorized_
plusTenPercent  (df.v))
```

[Useudf to define a DataFrame vudf.]

5.3.2.4 Grouped Vectorized UDFs (GVUDFs)

Grouped Vectorized UDFs (GVUDFs) use Panda library *split-apply-combine pattern* in data analysis. The GVUDF group function operates on all the data for a group, such as operate on all the data, "for each car showroom, compute yearly sales".

GVUDF steps are:

1. Splits a Spark DataFrame into groups based on the conditions specified in the groupby operator
2. Applies a vectorized user-defined function (pandas.DataFrame -> pandas.Dataframe) to each group
3. Combines into new group
4. Returns the results as a new Spark DataFrame

Pandas GVUDF, *pandas_GVUDF,* (i) uses the function similar to pandas_VUDF, (ii) pandas.DataFrame as input to the GVUDF, (ii) pandas.DataFrame as output from the GVUDF, (iii) grouping semantics defined using clause GroupBy, (iv) output size can be any

and can be grouped and can be distinct from input, and (v) returns data type is a StructType. The type defines a column name and the type of the returned pandas.DataFrame.

The following example explains GVUDF for adding 10% in a cost of group of rows for toy products.

EXAMPLE 5.6

Add 10% cost in each value of item cost in a group of rows. Use GVUDF to define a `DataFrame costTenPercetPlus`GVUDF.

SOLUTION

Following are the Python statements from pyspark.sql.functions import pandas_udf:

```
from pyspark.sql.functions  import pandas_udf
```

[Usepandas_udf to define a grouped vectorized udf.]

```
@pandas_udf(df.schema)
# Input/output are both a pandas.DataFrame def
costTenPercetPlus  (pdf):
return pdf.assign(v=add(pdf.v  +  O.lxpdf.v))
df.groupby('id') .apply(costTenPercetPlus)
```

5.3.3 Data Analysis Operations

Examples of operations required in the above analysis are given below:

1. Filtering single and multiple columns
2. Creating a top-ten list with values or percentages
3. Setting up sub-totals
4. Creating multiple-field criteria filters
5. Creating unique lists from repeating field data
6. Finding duplicate data with specialized arrays, and using the remove duplicates command and removing outliers
7. Multiple key sorting

8. Counting the number of unique items in a list
9. Using SUMIF and COUNTIF functions
10. Working with database functions, such as DSUM and DMAX
11. Converting lists to tables.

5.3.3.1 Removing *Outliers for Data Quality Improvement for Analysis*

Outliers are data which appear as they do not belong to the data set. The Outliers are generally results of human data-entry errors, programming bugs, some transition effect or phase lag in stabilizing the data value to the true value.

The actual outliers need to be removed from the data set. For example, missing decimal in the cost of a toy US$ 1.85 will make the cost 100 times more for a single toy. The result will thus be affected by a small or large amount. When valid data is identified as an outlier, then also the results are affected.

The statistical mean is computed from the product of each observed value v of *Values* with probability (or weight) P and then taking the average. The variance equals the difference of a value with respect to the mean, then square that, and then average the results. Standard deviation isjust the square root of the variance.

The following example explains the Python codes for removing outliers.

EXAMPLE 5.7

How will you remove outliers in values in a column?

SOLUTION

Transform ROD string or other to numeric data so that statistical methods compute and remove those who have larger distanceNumerics.

```
distanceNumerics
(string))

stats=  distanceNumerics.stats()
```

[stats() means a statistical function, such as meanl), stdev()]'

```
statdev = std.stdev() mean=  stats.mean() reasonableDistances
distanceNumerics.filter    (PySpark values:  maths.fabs   (values-
mean) <  3  xstdev)
```

[Assume that distances that are reasonable are less than three times the standard deviation. Distance means difference with respect to mean or peak value.]

```
print   reasonableDistances.collect()
```

[Print the values within reasonable distances.]

Self-Assessment Exercise linkedto LO 5.2

1. What are the steps between acquisition of data from different sources, applications of analyzed data, and applications support by Spark for analyses?
2. What are the different sources from which the Dataframes are created for query processing?
3. What are grouped vectorized UDFs? How do they differ from UDFs?
4. List the actions of the count(*), count(expr); sum(col), sum(DISTINCTcol), avglcol), avg(DISTINCTcol), min(col) and DOUBLE max(col) aggregation functions. 5. How is a four-column Dataframe created using a parquetFile?
6. List the actions of each statement in codes given in Example 5.7.

5.41 DOWNLOADING SPARK, AND PROGRAMMING USING RODS AND MLIB

The following subsections describe downloading of Spark, getting started in programming with RDDs and introduces Machine |gettDo• ffiirngl|oo.d st.aillilglted| 5 in,park .

learning with the MLib: 1progra m mi n.g1 \!l'ith Sp:a rk, Spair:k slnlel I, Spar:k context devel opillilg .amii tesmhmg1 the

5.4.1 Downloading, Installing Spark and Getting

codes, rp,rogrnrm rn illilg .vith tllile |R DUs. :allild| a ppli cations

Started

ofr, Lib

Spark 2.3.1 uses Scala 2.11.x API when using compatible Scala version 2.11.x. Initially select the choices for the download: (i) *Choose a Spark Release:* 2.3.1 (Iune, 2018), (ii) *Choose a package* type: pre-built for Hadoop 2.7 or later, (iii) *Choose a download type:* Direct Download, and (iv) *Verify this release using the* 1.2.0 signatures and checksums.

Spark new versions run onJava 8+, Python 2.7+/3.4+ and R 3.1+. Programmers using Scala

Text, Web Content,Link, and Social Network Analytics

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

LO 9.1 Use the methods of text mining and machine learning (ML)- Naive-Bayes classifier, and support vector machines for text analytics

LO 9. 2 Get knowledge and use the methods of mining the web-links, web-structure and web-contents, and analyzing the web graphs

LO 9.3 Get knowledge and use methods of PageRanking, analysis of web-structure, and discovering hubs, authorities and communities in web-structure

LO 9 .4 Get concepts representing social networks as graphs, social network analysis methods, finding the clustering in social network graphs, evaluating the SimRank, counting triangles (cliques) and discovering the communities

RECALL FROM EARLIER CHAPTERS

Graph Data Stores consist of various interconnected data nodes (Section 3.3.5). Models of graph and graph network organization describe the entities and objects, along with their relationships, associations and properties (Sections 8.2 and 8.3). Web and social• network graphs are examples of Graph network organization.

Graph structure analytics discovers the degree of interactions, closeness, betweenness, ranks, influences, probabilities distribution, beliefs and potentials. Analysis of the community and network discovers the *close-by* entities and fully mesh like connected sets. Network graph analyzes centralities, and computes the PageRank of the links (Section 8.4 and 8.5).

9.1 ! INTRODUCTION

Text Analytics often termed as 'text mining' refers to analyzing and extracting the meanings, patterns, correlations and structure hidden in unstructured and semi-structured textual data. Text data stores consist of strong temporal dimensions, have modularity over time and sources, such as topics and sentiments.

Methods of machine-learning are prevalent in text analytics also. For example, when a user books an air-flight ticket using a tablet or desktop, the user receives an SMS on the mobile about details of the booking and flight timings. An ML algorithm, such as *Windows Crotona* at the mobile reads and learns by itself from the SMSs received at the phone. *Crotona* uses the ML for the SMS text analysis .. Learning results in SMS alerts to the user. An alert is reminder a day before the flight. Another alert is two hours before the flight, about the need to reach the airport. Those alerts are system-generated without prior request from the user.

The reader is required to know the meaning of the following select key terms:

Vector refers to an entity with number of interrelated elements. For example, a data point consists of n-elements in an n• dimensional space, and represents a vector to that point from the origin in the space. A word is a vector of characters as the elements. Consider a vector representation of word 'McGraw-Hill', then vector vMH = [M, c, G, r, a, w, -, H, i, **1,** 1]. vMH is vector of 11 elements (characters) that refers to word McGraw-Hill.

Feature refers to a set of properties associated with an entity, object or category. For example, feature of properties, such as description of data analysis, data cleaning, data visualization and other topics in a book on data analytics.

Category refers to a classification on the basis of set of distinct features (for example, a category of text, document, cars, toys, students, news or fruits).

Label refers to a name assigned to a category, for example to sports-news, latest data analytics books.

Dimension refers to a number of associated values, features or states, along the distinct spaces (dimensions). For example, a sentence has a number of words, each word has a number of characters, each word may have a feature, and so the sentences are in a three-dimensional space. Two dimensions are in metric spaces, which mean values in quantifiable spaces, such as the number of words, probability of occurrences in sentences, etc. Third dimension is in feature space, measured by a feature such as noun, verb, adverb, preposition, punctuation marks and stop word.

Another example is *apples.* Metric space variables are values, such as variables n, *number of apples* of specific properties, and P the *probability of preferring* apples of specific properties. Assume, feature space variables are four properties, *colour, shape, type* and *freshness.* Metric parameters and properties of apples are said to be in six-dimensional space, two are metric space (n and P) and four are feature space.

Graph data model refers to the data modelled by a set of entities. The entities identify by vertices V. A set of relations or associations identifies by edges E. An edge *e* represents a relation or association between two entities. Nodes represent the entities in the graph. The model also represents a hierarchy between the parent and children nodes.

Graph data network organization refers to a structure created by organizing entities or objects in a network, such as social network, business network and student network. A network organization means where persons or entities interconnect with each other, and have areas of common interest, business or study. A graph enables ease in traversing from one entity, person or web page link to another in the network by following a path. Web graph and social network graph enable such analysis. A graph network organization models the web and social networks. Examples of social networks are SlideShare, Linkedln, Facebook and Twitter. The analytics of social networks finds the link ranks, clusters and correlations. The analytics discovers hubs and communities.

Web content mining refers to the discovery of useful information from web documents and services. Search engines use web content mining. A search provides the links of the required information to the user.

Hyperlinks refer to links mentioned in the contents that enable the retrieval of contents at web, file, object or resources repository.

Link analytics means web structure mining of hyperlinks between web documents. The analytics of links and analyzing them for metrics such as page ranks, clusters, correlations, hubs and communities.

Count triangles Algorithm is an algorithm that finds a number of triangular relationships among the nodes. Triangular relationships mean interrelations between each other.

Graph node centrality metric means the centrality of a node in reference to other nodes using certain metrics. Metrics used for centrality of a node are degree, closeness, betweenness or other characteristics of the node, such as rank, belief, potential, expectation, evidence, reputation or status.

Degree centrality of a node refers to the number of direct connections. Having more number of direct connections is not always a better metric. Better measure is the fact that the connection directs to significant results and tell how the nodes connect to the isolated node.

Betweenness centrality is a measure that provides the extent to which a node lies on paths between other nodes. A node with high betweenness signifies high influence over what flows in the network indicating importance of link and single point of failure.

Closeness centrality is the degree to which a node is near all other nodes in a network (directly or indirectly). It reflects the ability to access information through the network.

The present Chapter focuses on text, web, contents, structure and social network graph analytics. Section 9.2 describes text mining and usage of ML techniques=Naive-Bayes analysis and support vector machines (SVMs) for analyzing text. Section 9.3 describes web mining, methods to implement the system, and analyzing the web graphs.

Section 9.4 describes PageRank methods, web structure analytics and finding the hubs and communities. Section 9.5 describes social network analysis, representation of social networks as graphs and computational methods of finding the clustering in social network graphs, evaluating the SimRank, counting triangles (cliques) and discovering the communities.

This chapter follows a method of notations as mentioned earlier in Section 6.1 for fonts when absolute value, mean value, function value, vector element or set member, entity or variable when these denote by a character or character-set. This chapter follows the notations for the probabilities, earlier specified in
Section 8.3.2. Condition probability P specifies as P (xilck) which means probability of variable x = xi at condition c = ck.

9.2 ! TEXT MINING

LO 9.1

i.thods of text mining and machinele;uning(L)• Naive-Bayesdassifier,and support vectormachines for text analytics

Today, large amounts of textual data is generated in computing applications. Text stream arriving continuously over time generates text data. For example, news articles, news reports, online comments on news, online traffic reports, corporate reports, web searches, and contents at social media discussion• forums (such as Linkedin, Twitter and Facebook), short messages on phones, chat messages, transcripts of phone conversations, biogs and e-mails.

The abundance of textual data leads to problems which relate to their collection, exploration and ways of leveraging data. Textual data presents challenges for computing and storage requirements, consists of a strong temporal dimension, has modularity over time and have sources such as topics and sentiments. Examples of text processing techniques are clustering analysis, classifications, evolution analysis and event detection. Following subsections describe text mining in details:

9.2.1 Text Mining

Four definitions are:

1. "Text mining refers to the process of deriving high-quality information from text." (Wikipedia)

2. "Text mining is the process of discovering and extracting knowledge from unstructured data." (National Center of Text Mining

   -The University of Manchester+)

3. "Text mining is the process of analyzing collections of textual contents in order to capture key concepts themes, uncover hidden relationships, and discover the trends without requiring that you know the precise words or terms that authors have

   used to express those concepts." (IBM2)

4. "Text mining is a technique which helps in revealing the patterns and relationships in large volumes of textual content that are not visible to the naked eye, leading to new business opportunities and improvements in processes." (Amazon BigData Official

   Blog3)

Applications of text mining in business domains are predicting stock movements from analysis of company results, decision making for product and innovations developed at the company and contextual advertising. Some other applications are (i) mail filtering (spam), (ii) drug action reports (iii) fraud detection (iv) knowledge management, and (iv) social media data analysis.

The applications provide innovative and insightful results. The results when combined with other data sources, find the answers to the following:

(i) Two terms which occur together

(ii) Information linkage with another information

(iii) Different categories that can be created from extracted information (iv) Prediction of information or categories.

9.2.1.1 Text Mining Overview

Text mining includes extraction of high-quality information, discovering and extracting knowledge, and revealing patterns and relationships from unstructured data available in the form of text.

The term *text analytics* evolves from provisioning of strong integration with the already existing database technology, artificial intelligence, machine learning, data mining and text Data Store techniques. Information retrieval, natural language processing (NLP), classification, clustering and knowledge management are some of such useful techniques. Figure 9.1 shows process-pipeline in text• analytics.

9.2.1.2 Areas and Applications of Text Mining

Natural Language Processing (NLP) is a technique for analyzing, understanding and deriving meaning from human language. NLP involves the computer's understanding and manipulation of human language. NLP algorithms are typically based on ML algorithms. They automatically learn the rules. First, they analyze set of examples from a large collection of sentences in a book. Then, they make the statistical inferences.

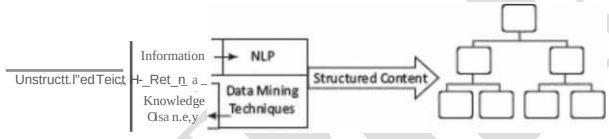


Figure 9.1 Text analytics process pipeline

NLP contributes to the field of human computer interaction by enabling several real-world applications such as automatic text summarization, sentiment analysis, topic extraction, named entity recognition, parts-of-speech tagging, relationship extraction and stemming. The common uses of NLP include text mining, machine translation and automated question answering.

Information Retrieval (IR) is a process of searching and retrieving a subset of documents from the abundant collection of documents. IR can also be defined as extraction of information required by a user. IR is an area derived fundamentally from database technology. One of the most popular applications of IR is searching the information on the web. Search engines provide IR using various advance techniques. For example, the crawler program is capable of retrieving information from a wide variety of data sources. Search methods use metadata or full-text indexing.

Information Extraction (IE) is a process in which the software extracts structured information from unstructured and/or semi• structured documents. IE finds the relationship within text or desired contents from text. IE ideally derives from machine learning, more specifically from the NLP domain. Content extraction from the images, audio or video is an example of information extraction.

IE requires a dictionary of extraction patterns (For example, "Citizen of <x>, or "Located in -oc-") and a semantic lexicon (dictionary of words with semantic category labels).

Document Clustering is an application which groups text documents into clusters. Automating document organization, topic extraction and fast information retrieval or filtering use the document clustering method. For example, web document clustering facilitates easy search by users.

Document Classification is an application to classify text documents into classes or categories. The application is useful for publishers, news sites, biogs or areas where lot of contents are present.

Web Mining is an application of data mining techniques. They discover patterns from the web Data Store. The patterns facilitate understanding. They improve the services of web-based applications. Data mining of web usage provides the browsing behavior of a website.

Concept Extraction is an application that deals with the extraction of concept from textual data. Concept extraction is an area of text classification in which words and phrases are classified into a semantically similar group.

9.2.1.3 Text Mining Process

Text is most commonly used for information exchange. Unlike data stored in databases, text is unstructured, ambiguous and difficult to process. Text mining is the process that analyzes a text to extract information useful for a specific purpose.

Syntactically, a text document comprises characters that form words, which can be further combined to generate phrases or sentences. Text mining steps are (i) recognizing, extracting and using the information present in words. Along with searching of words, mining involves search for semantic patterns as well.

Text mining process consists of a process-pipeline. The pipeline processes execute in several phases. Mining uses the iterative and interactive processes. The processing in pipeline does text mining efficiently and mines the new information. Figure 9.2 shows five phases of the process pipeline.

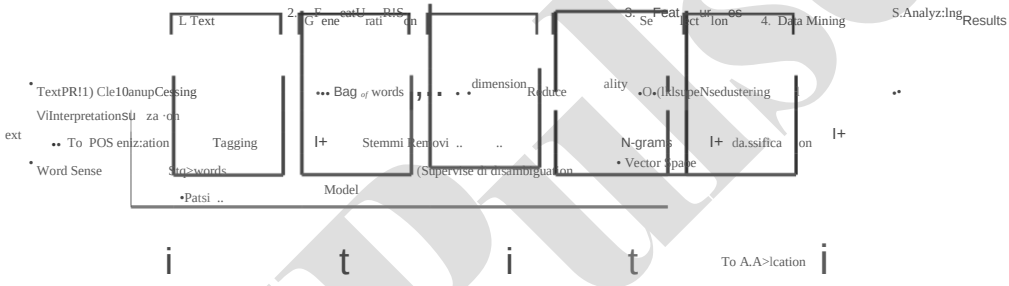


Figure 9.2 Five phases in a process pipeline

The following subsection describes these phases:

9.2.1.4 Text Mining Process Phases

The five phases for processing text are as follows:

Phase 1: Text pre-processing enables Syntactic/Semantic text-analysis and does the followings:

1. Text *cleanup* is a process of removing unnecessary or unwanted information. Text cleanup converts the raw data by filling up the missing values, identifies and removes outliers, and resolves the inconsistencies. For example, removing comments, removing or escaping "%20" from URL for the web pages or cleanup the typing error, such as teh (the), don't (do not) [%20 specifies space in a URL].
2. *Tokenization* is a process of splitting the cleanup text into tokens (words) using white spaces and punctuation marks as delimiters.
3. *Part of Speech (POS) tagging* is a method that attempts labeling of each token (word) with an appropriate POS. Tagging helps in recognizing names of people, places, organizations and titles. English language set includes the noun, verb, adverb, adjective, prepositions and conjunctions. Part of Speech encoded in the annotation system of the Penn Treebank Project has 36 POS tags. [4]
4. *Word sense disambiguation* is a method, which identifies the sense of a word used in a sentence; that gives meaning in case the word has multiple meanings. The methods, which resolve the ambiguity of words can be context or proximity based. Some examples of such words are bear, bank, cell and bass.
5. *Parsing* is a method, which generates a parse-tree for each sentence. Parsing attempts and infers the precise grammatical relationships between different words in a given sentence.

Phase 2: Features Generation is a process which first defines features (variables, predictors). Some of the ways of feature generations are:

1. *Bag of* words-Order of words is not that important for certain applications.

   Text document is represented by the words it contains (and their occurrences). Document classification methods commonly use the bag-of-words model. The pre-processing of a document first provides a document with a bag of words. Document classification

methods then use the occurrence (frequency) of each word as a feature for training a classifier. Algorithms do not directly apply on the bag of words, but use the frequencies.

2. Stemming-identifies a word by its root.

   (i) Normalizes or unifies variations of the same concept, such as *speak* for three variations, i.e., speaking, speaks, speakers denoted by [speaking, speaks, speaker ---+ speak]

   (ii) Removes plurals, normalizes verb tenses and remove affixes.

   Stemming reduces the word to its most basic element. For example, impuriflcation r+ pure.

3. *Removing stop words* from the feature space-they are the common words, unlikely to help text mining. The search program tries to ignore stop words. For example, ignores *a, at, for,* it, *in* and *are.*

4. *Vector Space Model* (VSM)-is an algebraic model for representing text documents as vector of identifiers, word frequencies or terms in the document index. VSM uses the method of term frequency-inverse document frequency (TF-IDF) and evaluates how important is a word in a document.

   When used in document classification, VSM also refers to the bag-of-words model. This bag of words is required to be converted into a term-vector in VSM. The term vector provides the numeric values corresponding to each term appearing in a document. The term vector is very helpful in feature generation and selection.

Term frequency and inverse document frequency (IDF) are important metrics in text analysis. TF-IDF weighting is most common• Instead of the simple TF, IDF is used to weight the importance of word in the document.

TF-IDF stands for the 'term frequency-inverse document frequency'. It is a numeric measure used to score the importance of a word in a document based on how often the word appears in that document and in a given collection of documents. It suggests that if a word appears frequently in a document, then it is an important word, and should therefore be high in score. But if a word appears in many more other documents, it is probably not a unique identifier, and therefore should be assigned a lower score. The TF-IDF is measured as:

TF-IDF<t> No. of times t appears in a document x log No. of documents in the collection (9 .1)
Total No. of terms in the document No. of documents that contain t where t denotes the term vector.

Following example suggests method of calculating TF-IDF (t).

EXAMPLE 9.1

Consider a document containing 1000 words wherein the word *toys* appears 16 times. How will the TF-IDF weight be calculated?

SOLUTION

The term frequency (TF) for *toys* is then (16/1000) = 0.016. Let, there are 10 million documents and the word *toys* appear in 1000 of them. Then, the inverse document frequency (IDF) is calculated as $\log_{10}$ (10,000,000/1,000) = 4.

TF-IDF weight= 0.016 x 4 = 0.064

Additional weight is assigned to terms appearing as keywords or in titles. Documents are usually represented as a sparse vector of terms weights and extra weights are added to the terms appearing in title or keywords.

Pre-processing of web data succeeds the conversion of bag of words into vector space model (VSM) or simply by vector creation.

Common Information Retrieval Technique - Vector space model (VSM) is an algebraic model for representing textual information as vectors of identifiers, such as, index terms. Information retrieval methods use VSM.

Each document or HTML page represents by a sparse vector of term weights. The sparse matrices represent the term frequencies (TFs).

(Sparse vector and sparse-matrix have many elements as zero or null. An associated metadata enables data storing of them in a form which does not include zeros in case of large datasets. The metadata then includes indices map with the positions in the list of elements of the vector or matrix.)

The following example gives the conversion method for evaluating TFs and matrix in pre-processing phase.

EXAMPLE 9.2

Assume that the documents below define the document space with five documents di, dz, d3, d4 and ds:

Train Document Set:

dt: Children like the toys.

dz: The toys are precious.

Test Document Set:

d3: There are many toys in the shop.

d4: Some toys are precious and some toys are costly as well.

ds: The toys shop is one of the famous shops.

How will be the documents term vector and matrix be calculated for features generation/selection?

SOLUTION

First, create an index vocabulary of the words of the train document set using the documents dt and dz from the document set. The index vocabulary E(t) where tis the term will be:

$$Eu) = \begin{cases} 1.\ \text{when } t = \text{"children"} \\ 2.\ \text{when } t = \text{"toys"} \\ 3.\ \text{when } t = \text{"precious"} \\ 4.\ \text{when } t = \text{"shop"} \\ 5.\ \text{when } t = \text{"costly"} \\ 6.\ \text{when } t = \text{"famous"} \end{cases}$$

Note that the stop words are already not considered during the pre-processing step. The term frequency (TF) is a measure of how many times the terms present in vocabulary E(t) are present in the documents ds, d4 and ds.

$$TF(l.d) = \underset{t \in J}{I,\text{count}}\ (x.t) \qquad (9.2)$$

where the count (x, t), is a simple function defined as:

$$\text{coumt.r.r} = \begin{cases} 1. & \text{if } x = t \\ 0. & \text{otherwise} \end{cases} \quad (9.3)$$

For example, TF ("toys", ds) = 2.

Create the document vector as:

$$v_{1,f} = <TFul.dn).TFU2.dn) \quad TF(tn.dnJ) \quad (9.4)$$

Thus, the documents d3, d4 and ds are represented as vector as:

$$v_{43} = <TFUl.d3J.TF(r2.d3) \quad TFUn.d3))$$
$$v_{,14} = (TF<tl.d4).TF(t2.d4J \quad TF(tn.d4))$$
$$v_{45} = (TFU1.d5).TF<t2.d5)\cdot\ldots. \quad TFUn.d5>I \quad (9.5)$$

This gives:

$$I'_{43} = (I.I.0.1.0.0)$$
$$v_{44} = (0,2.L0.1.01$$
$$I'_{45} = (0.1.0.2.0.1) \quad (96)$$

The resulting vector $\dot{I}'_{,3}$ shows 1 occurrence of the term "children", 1 occurrence of the term "toys" and so on. In the $i\text{-}14$, there is 0 occurrence of the term "children", 2 occurrences of the term "toys" and so on.

A collection of web documents requires representation as vectors. Another representation is a matrix with IDI x F shape, where IDI is the cardinality of the document space (total number of documents) and the Fis the number of features. F represents the vocabulary size in the example. Matrix representation of the vectors described above is by a 3 x 6 matrix as follows:

$$MIDIXF = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 & 0 & 1 \end{bmatrix} \quad (9.7)$$

Example 9.2 shows that the matrices representing term frequencies tend to be very sparse (with majority of terms zeroed). A common representation of such matrix is thus the sparse matrices.

Phase 3: Features Selection is the process that selects a subset of features by rejecting irrelevant and/or redundant features (variables, predictors or dimension) according to defined criteria. Feature selection process does the following:

1. *Dimensionality* reduction-Feature selection is one of the methods of division and therefore, dimension reduction. The basic objective is to eliminate irrelevant and redundant data. Redundant features are those, which provide no extra information. Irrelevant features provide no useful or relevant information in any context.

   Principal Component Analysis (PCA) and Linear Discriminate Analysis (LDA) are dimension reduction methods. Discrimination ability of a feature measures relevancy of features. Correlation helps in finding the redundancy of the feature. Two features are redundant to each other if their values correlate with each other.

2. *N-gram* evaluation-finding the number of consecutive words of interest and extract them. For example, 2-gram is a two words sequence, ["tasty food", "Good one"]. 3-gram is a three words sequence, ["Crime Investigation Department"].

3. Noise *detection and evaluation of* outliers methods do the identification of unusual or suspicious items, events or observations from the data set. This step helps in cleaning the data.

The feature selection algorithm reduces dimensionality that not only improves the performance of learning algorithm but also reduces the storage requirement for a dataset. The process enhances data understanding and its visualization.

Phase 4: Data mining techniques enable insights about the structured database that resulted from the previous phases. Examples of techniques are:

1. Unsupervised learning (for example, clustering)
   (i) The class labels (categories) of training data are unknown
   (ii) Establish the existence of groups or clusters in the data

Good clustering methods use high intra-cluster similarity and low inter-cluster similarity. Examples of uses - biogs, patterns and trends.

2. *Supervised learning (for example, classification)*

(i) The training data is labeled indicating the class

(ii) New data is classified based on the training set

Classification is correct when the known label of test sample is identical with the resulting class computed from the classification model.

Examples of uses are *news filtering application,* where it is required to automatically assign incoming documents to pre-defined categories; *email spam filtering,* where it is identified whether incoming email messages are spam or not.

Example of text classification methods are *Naive Bayes Classifier* and *SVMs.*

3. *Identifying evolutionary patterns* in temporal text streams-the method is useful in a wide range of applications, such as summarizing of events in news articles and extracting the research trends in the scientific literature.

Phase 5: Analysing results

(i) Evaluate the outcome of the complete process.

(ii) Interpretation of Result- If acceptable then results obtained can be used as an input for next set of sequences. Else, the result can be discarded, and try to understand what and why the process failed.

(iii) Visualization - Prepare visuals from data, and build a prototype.

(iv) Use the results for further improvement in activities at the enterprise, industry or institution.

Open source tools, such as *nltk* are available for text analytics. Online contents accompanying book describe how text analytics tasks can be performed using Python library *nltk* in the solution of Practice Exercise 9.2.

9.2.1.5 Text Mining Challenges

The challenges in the area of text mining can be classified on the basis of documents area-characteristics. Some of the classifications are as follows:

1. NLP issues:
   (i) POS Tagging
   (ii) Ambiguity
   (iii) Tokenization
   (iv) Parsing
   (v) Stemming
   (vi) Synonymy and polysemy
2. Mining techniques:
   (i) Identification of the suitable algorithm(s)
   (ii) Massive amount of data and annotated corpora
   (iii) Concepts and semantic relations extraction
   (iv) When no training data is available

3. Variety of data:

   (i) Different data sources require different approaches and different areas of expertise

   (ii) Unstructured and language independency

4. Information visualization

5. Efficiency when processing real-time text stream

6. Scalability

9.2.1.6 Supervised Text Classification

The categorization of text documents requires information retrieval, ML and NLP techniques. Some important approaches to automatic text categorization are based on ML techniques.

The *supervised text classification* requires labeled documents and additional knowledge from experts. The algorithms exploit the training data (where zero or more categories) to learn a classifier, which classifies new text documents and labels each document. A document is considered as a positive example for all categories with which it is labeled, and as a negative example to all others. The task of a training algorithm for a text classifier is to find a weight vector which best classifies new text documents.

The different approaches for supervised text classification are:

(i) K-Nearest Neighbour Method

(ii) Support Vector Machine

(iii) Naive Bayes Method

(iv) Decision Tree

(v) Decision Rule

K-Nearest Neighbours (KNN) method makes use of training text document. The training documents are the previously categorized set of documents. They train the system to understand each category. The classifier uses the training 'model' to classify new incoming documents. KNN assumes that close-by objects are more probable in the same category. KNN finds k objects in the large number of text documents, which have most similar query responses. Thus, in KNN, predictions are based on a method that is used to predict new (not observed earlier) text data. The predictions are by (i) majority vote method (for classification tasks) and (ii) averaging (for regression) method over a set of K-nearest examples.

The decision trees or decision rules are built to predict the category for an input document. A decision tree or rule represents a set of nested logical if-then conditions on the observed values of the text features that enable the prediction of the target variable. The decision tree and decision rules are also used to classify (categorize) the document. Classification is done by recursively splitting the text features into a set of non-overlapping regions (Refer Section 6.8). (Section 6.7.4)

The following subsections describe Naive Bayes Method and Support Vector Machines in detail.

9.2.2 Naive Bayes Analysis

Naive Bayes classifier is a simple, probabilistic and statistical classifier. It is one of the most basic text classification techniques, also known as multivariate Bernoulli method. Naive Bayes classifies using Bayes theorem along with the Naive independence assumptions (conditional independence). The classifier computes condition probabilities for the conditional independence (Refer Section 8.3.2).

Probability that a bag-of-words ~ belong to kth class equals the product of individual probabilities of those words. P (~lck) = II P (xilck), where xi is a discrete random variable (word), i = 1, 2, ..., n, where n is number of words in the bag. II is sign for the product of n terms. [P~lck) means probability of condition that state the value= xi and of c = ck (Example 8.6)].

The P <xlck) is normalized as all distributed probabilities equals 1. P <xlck) is normalized by dividing the product on right hand side by Li P (~lck) P (ck).

The following example gives the method of deciding the most likely class.

EXAMPLE 9.3

How is "maximum a posteriori (MAP)" used to obtain the most likely class and take a decision?

Text classification problem uses the words (or tokens) of the document in order to classify it on the appropriate class. Bayes' rule is applied to documents and classes. For a document (d) and class (c), we get:

$$Pie|d,= \frac{P(d|c,P(c)}{P<d)} \qquad (98;$$

The "maximum a posteriori (MAP)" (to obtain most likely class) decision rule is applied to documents and classes:

(99)

SOLUTION

(MAP is "maximum posteriori" = most likely class)

$$c_{MAP} = \text{argma..,i;,""EC}\left(\frac{P(J!c)P(c))}{P(d)}\right) \quad \text{(Bayes Rule)} \tag{9.10}$$

oping the denominator) (9.11)

(9.12)

where t_1, t_2, ..., t_n are tokens of document.

Multinomial Naive Bayes independence assumptions

$$P(t_1.t_2 \ldots . t_n | c) \tag{9.13}$$

Bag-of-words assumption: Assume the position of the word does not matter.

Conditional independence: Assume the feature probabilities $P(t_i | c)$, are independent given the class(c):

$$P<t..,2 \cdots ,,,|e>= P(t\ \text{Id} \bullet P t t_2 \text{le}) \bullet P(t_3 \text{le}) \bullet \ldots \bullet P t,. \text{Ir}) \tag{9.14}$$

and thus, conditional independences are given by

$$c_{MAP} = \text{argmax}""\text{c} \quad (P(t..,t_2 \ldots . t, \text{lc}) P(\text{c})) \tag{9.15n}$$

$$c_{NB} = \text{argmax}_{c \in C}\left(P(c_j)\prod_{i \in T} P(t|c)\right) \tag{9.15b}$$

Applying multinomial Naive Bayes classifier to text classification where positions = all word positions in the text document,

$$cN8=argnuuc|ec(P(ci) \prod_{\bullet EPI>.tallOM} P(tih)) \tag{9.16}$$

The equation estimates the product of the probability of each word of the document given a particular class (likelihood), multiplied by the probability of the particular class (prior) to find in which class one should classify a new document.

Select the one with the highest probability among all the classes of set C. Calculation of product of the probabilities leads to float point underflow when handling numbers with specific decimal point accuracy by computing devices. Such small numbers will be rounded to zero, implying the analysis is of no use at all. In order to avoid this, instead of maximizing the product of the probabilities, the maximization of the sum of their logarithms is done:

$$c_{NB} = \text{argmax}_{c_j \in C}\left(\log P(c_j) \prod_{i \in \text{Positions}} \log P(t_i | c_j)\right) \tag{9.17}$$

Here, choose the one with the highest log score rather than choosing the class with the highest probability. Given that the logarithm function is monotonic, the decision of MAP remains the same.

When compared with other techniques, such as Random Forest, Max Entropy and SVM, the Naive Bayes classifier performs efficiently in terms of less CPU and memory consumption. Naive Bayes classifier requires a small amount of training data to estimate the parameters. The classifier is not sensitive to irrelevant features as well. Furthermore, the training time is significantly smaller with Naive Bayes as opposed to other techniques.

The classifier is popularly used in a variety of applications, such as email spam detection, personal email sorting, document categorization, language detection, authorship identification, age/gender identification and sentiment detection.

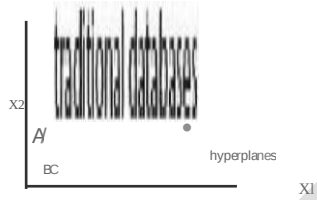
9.2.3 Support Vector Machines

Support vector machines (SVM) is a set of related *supervised learning methods* (the presence of training data) that analyze data, recognize patterns, classify text, recognize hand-written characters, classify images, as well as bioinformatics and bio sequence analysis.

A vector has in general n components, x2, x3, ... , xn. A datapoint represents by (Xl, X2, ... , Xn) in n-dimensional space. Assume for the sake of simplicity, that a vector has two components, Xl and X2 (Two sets of words in text analysis).

Section 6. 7 .6 described the use of the concept of hyperplanes for classification. A *hyperplane* is a subspace of one dimension less than its ambient space in geometry (Figure 6.18). If a space is 3-dimensional then its hyperplanes are 2-dimensional planes, while if the space is 2-dimensional, its hyperplanes are I-dimensional, which means lines.

The hyperplane which separates the two classes most appropriately has maximum distance from closest data points of the distinct classes. This distance is termed as margin. Figure 9.3 shows the concept of support vectors, separating hyperplane and margins when using Bas a classifier. The margin for hyperplane Bin Figure 9.3 is more as compared to two hyperplanes, A and C shown by dotted lines. The margin of the data points from B is maximum. Therefore, the hyperplane B is the maximum margin classifier.



M axi um margi7 The encircled I

I data points are I L~~~~~~~o~J

~patatin

Figure 9. 3 Support vectors, separating hyperplane (B) and margins

A and Care closest (least margins) to the data points. These are called the *support vectors*. They support the classifications of the star and dotted data points. [Remember that with n-dimensional datapoints space, a hyperplane has the vectors along (n - 1) axes.]

The support vectors are such that a set of data points lies closest to the decision (classification) surface (or hyperplane). Those points are most difficult to classify. They have direct bearing on the optimum location of the classification surface. Support vectors along maximum margin classification surface are thus gives the best results.

Thus, a SVM classifier is a *discriminative classifier* formally defined by a separating hyperplane. The concept applies extensively in number of application areas of ML. Applications of SVMs are as follows:

1. classification based on the outputs taking discrete values in a set of possible categories, SVM can be used to separate or predict if something belongs to a particular class or category. SVM helps in finding a decision boundary between two categories.
2. Regression analysis, if learning problem has continuous real-valued output (continuous values of x, in place discrete n values, (X1, X2, X3, ... , x.)
3. Pattern recognition 4. Outliers detection.

The following example illustrates the discriminative classifier method, formally defined by a separating a hyperplane for taking the decision for effective elements (entities, set of words, itemsets) in a training set.

EXAMPLE 9.4

How is the discriminative classifier used?

SOLUTION

Consider a mapping function (f) used for linear separation in the feature space (H). An optimal separating hyperplane depends on the data through dot products in H (f(xJf(J(_j)),

A kernel function k is required that acts as a measure of similarity. Let us use k where

LO 4.3

(9.1)

The objective is to select the hyperplane which separates the two classes most appropriately. This helps in identifying the right or appropriate hyperplane. Figure 9.3 showed three hyperplanes. A, B and C. A and C are least margin planes and B is maximum margin plane.

A hyperplane,maximum margin classifier is the right hyperplane. It has maximum distances from the nearest data points (of either classes). An important reason for selecting the maximum margin classification surface is robustness. A hyperplane having low margin has considerably high chance of misclassifying.

Binary Classification

For a given training data [xi, y(xi)] for i = 1 ...N, with xi E !Rd and Yi E {-1,1},learn a classifier f(x) such that:

$$f(x_i) \begin{cases} > -c & y_i = +1 \\ < -c & y_i = -1 \end{cases} \quad \text{or equivalently} \quad y_i f(x_i) > c \tag{9.19}$$

The above equation implies that $y_i f(x_i) > c$ for correct classification.

Figure 9.4 shows a two-class classification. The method is using one hyperplane B for separating two-class classification of data points.

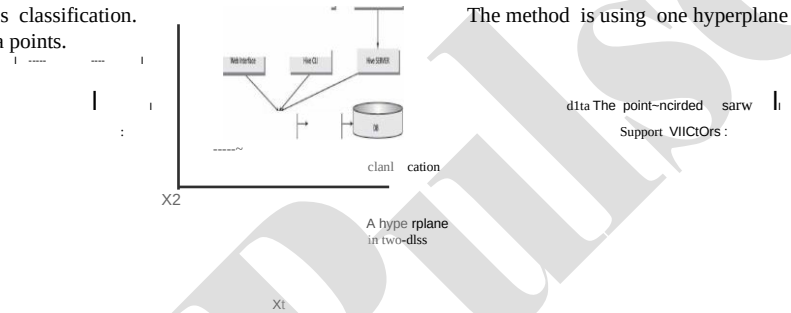


Figure 9. 4 Concept of training set data using support vectors

Let us take the simplest case of two-class classification. Suppose there are two features Xl and X2 and it is required to classify objects as shown in the Figure 9.4. Stars and dots represent the objects (itemsets, sets of words, entities) of two classes. The goal is to design a hyperplane (B) that classify all training vectors in two classes for linearly separable binary set.

The following example gives the method to design a hyperplane (B) that classify all training vectors:

EXAMPLE 9.5

How will you select an appropriate hyperplane that classifies all training vectors?

SOLUTION

Plane Bis f(x) = wx + b where w is weight vector. Figure 9.5 shows the method of selecting the right hyperplane.

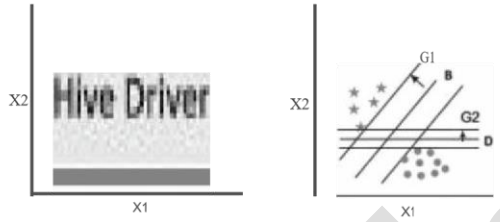


Figure 9.5 Method of selecting the right hyperplane

The best choice is the hyperplane that leaves the maximum margin from both the classes. Thus, Bis the right choice, since G1 > G2.

Thus, for line segment B,

$$f(x) \sim 1 \text{ "/ e } x \text{ -+ class } 1$$
$$f(x) \sim -1 \text{ "/ e } x \text{ -+ class } 2 \qquad (9.20)$$

Self-Assessment Exercise linked to LO 9.1

1. Define text analytics.
2. List the steps in text pre-processing phase. Why are tokenization and POS tagging needed? Give an example of each step.
3. How is bag-of-word used in text analysis? Give 5 examples of stemming the affix wordforms to its root word.
4. How do the TF-IDF weighting and sparse matrices represent the term frequencies (TFs) for use in text analysis?
5. How does maximum a posteriori (MAP) in Naive Bayes classifier enable the decision about classification?
6. How does support vectors in SVMsclassify the data points in n-dimensional space.

9.31 WEB MINING, WEB CONTENT AND WEB USAGE ANALYTICS

LO 9.2

1i n lng the veb<Il n ks, vebstructure and web• contents. and analyzing the vebgr.iphs

Web is a collection of interrelated files at web servers. Web data refers to (i) web content-text, image and records, (ii) web structure-hyperlinks and tags, and (iii) web usage-http logs and application server logs.

Features of web data are:

1. Volume of information and its ready availability
2. Heterogeneity
3. Variety and diversity (Information on almost every topic is available using different forms, such as text, structured tables and lists, images, audio and video.)
4. Mostly semi-structured due to the nested structure of HTMLcode
5. Hyperlinks among pages within a website, and across different websites

6. Redundant or similar information may be present in several pages
7. Mostly, the web page has multiple sections (divisions), such as main contents of the page, advertisements, navigation panels, common menu for all the pages of a website and copyright notices
8. A web form or HTMLform on a web page enables a user to enter data that is sent to a server for processing
9. Website contents are dynamic in nature where information on the web pages constantly changes, and fast information growth takes place such as conversations between users, social media, etc. The following subsections describe web data mining and analysis methods:

9.3.1 Web Mining

Data Mining is a process of discovering patterns in large datasets to gain knowledge. The process can be shown as [Raw Data - Patterns - Knowledge]. Web data mining is the mining of web data. Web mining methods are in multidisciplinary domains: (i) data mining, ML, natural language, (ii) processing, statistics, databases, information retrieval, and (iii) multimedia and visualization.

Web consists of rich features and patterns. A challenging task is retrieving interesting content and discovering knowledge from web data. Web offers several opportunities and challenges to data mining.

Definition of Web Mining

Web mining refers to the use of techniques and algorithms that extract knowledge from the web data available in the form of web documents and services. Weh mining applications are as follows:

(i) Extracting the fragment from a web document that represents the full web document

(ii) Identifying interesting graph patterns or pre-processing the whole web graph to come up with metrics, such as PageRank

(iii) User identification, session creation, malicious activity detection and filtering, and extracting usage path patterns

Web Mining Taxonomy

Web mining can broadly be classified into three categories, based on the types of web data to be mined. Three ways are web content mining, web structure mining and web usage mining. Figure 9.6 shows the taxonomy of web mining.

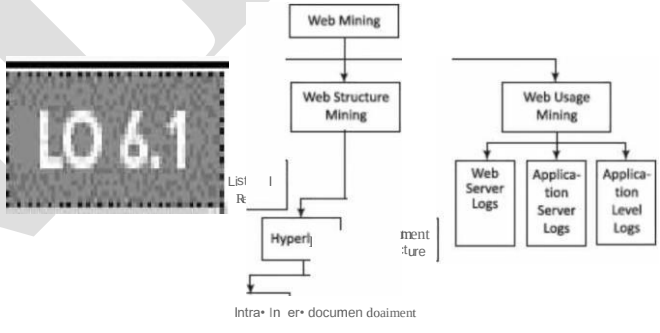


Figure 9.6 Web mining taxonomy

Web content mining is the process of extracting useful information from the contents of web documents. The content may consist of text, images, audio, video or structured records, such as lists and tables.

Web structure mining is the process of discovering structure information from the web. Based on the kind of structure-information present in the web resources, web structure mining can be divided into:

1. Hyperlinks: the structure that connects a location at a web page to a different location, either within the same web page (intra• document hyperlink) or on a different web page (inter-document hyperlink)

2. Document Structure: The structure of a typical web graph consists of web pages as nodes, and hyper links as edges connecting the related pages.

Web usage mining is the application of data mining techniques which discover interesting usage patterns from web usage data. The data contains the identity or origin of web users along with their browsing behavior at a web site. Web usage mining can be classified as:

(i) Web Server logs: Collected by the web server and typically include IP address, page reference and access time.

(ii) Application Server Logs: Application servers typically maintain their own logging and these logs can be helpful in troubleshooting problems with services.

(iii) Application Level Logs: Recording events usually by application software in a certain scope in order to provide an audit trail that can be used to understand the activity of the system and to diagnose problems.

9.3.2 Web ContentMining

Web ContentMining is the process of information or resource discovery from the content of web documents across the World Wide Web. Web content mining can be (i) direct mining of the contents of documents or (ii) mining through search engines. They search fast compared to direct method.

Web content mining relates to both, data mining as well as text mining. Following are the reasons:

(i) The content from web is similar to the contents obtained from database, file system or through any other mean. Thus, available data mining techniques can be applied to the web.

(ii) Content mining relates to text mining because much of the web content comprises texts.

(iii) Web data are mainly semi-structured and/or unstructured, while data mining is structured and the text is unstructured.

Applications

Following are the applications of content mining from web documents:

1. Classifying the web documents into categories
2. Identifying topics of web documents
3. Finding similar web pages across the different web servers
4. Applications related to relevance:

   (a) Recommendations - List of top "n" relevant documents in a collection or portion of a collection
   (b) Filters - Show/Hide documents based on some criterion

   (c) Queries - Enhance standard query relevance with user, role, and/or task-based relevance.

9.3.2.1 Common Web Content Mining Techniques

Pre-processing of contents The pre-processing steps are quite similar to the pre-processing for text mining. The content preparation involves:

1. Extraction of text from HTML
2. Data cleaning by filling up the missing values and smoothing the noisy data
3. *Tokenizing:* Generates the tokens of words from the cleaned up text
4. *Stemming:* Reduce the words to their roots. The different grammatical forms or declinations of verbs identify and index (count) as the same word. For example, stemming will ensure that both "closed" and "closing" are derived from the same word "close". Stemming algorithm, *Porter*, can be used here. The java code for *Porter* stemming algorithm can be obtained from *https://tartarus.org/martin/PorterStemmer /.java.bct,*

5. *Removing the stop* words: The common words unlikely to help in the mining process such as articles (a, an, the), or prepositions (such as, to, in, for) are removed.

6. *Calculate collection wide-word frequencies:* The distinct-word stem obtained after stemming process and removing the stop words results into a list of significant words (or terms). Calculating the occurrence of a significant term (t) in a collection is called collection frequency (CFt). CF counts the multiple occurrences.)

   Now, find the number of documents in the collection that contains the specific term (t). This numeric measure is the document frequency (DF t).

7. *Calculate per Document Term Frequencies* (TF). TF is a numeric measure that is used to score the importance of a word in a document based on how often it appeared in that document (Refer Example 9.1).

8. *Bag of* words: Web document is represented by the words it contains (and their occurrences.

The following example explains the concept of CF and DF using the data of toy sales collection.

EXAMPLE 9.6

Using the table below on collection and document frequencies, which is prepared from the toys sales collection, analyze the numeric values of CFs and DFs.

dJscoaot ISSSS SSO

Collection frequency (CF) and document frequency (DF)

lIle 1,'67 1230

SOLUTION

The table suggests that the collection frequency (CF) and document frequency (DF) can behave differently. The CF values for both *discount* and *sale* are nearly equal, but their DF values differ significantly. The reason is that the word sale is present in a large number of documents and the word discount in a less number of documents. Thus, when a query related to *discount* is generated, it must be searched in the concerned documents only.

Mining Tasks for Web Content Analytics

Following are the tasks for web content analytics: 1. classification - A supervised technique which: (i) Identifies the class or category a new web documents belongs to from the set of predefined classes or categories

(ii) Categories in the form of a term vector that are produced during a "training" phase

(iii) Employs algorithms using term vector to categorize the new data according to the observations at the training set.

2. *Clustering* - An unsupervised technique:

(i) Groups the web documents (clustered) with similar features using some similarity measure

(ii) Uses no pre-defined perception of what the groups should be

(iii) Measures most common similarity using the dot product between two web document vectors.

3. *Identifying the association* between web documents - Association rules help to identify correlation between web pages that occur mostly together.

The other significant mining tasks are:

1. *Topic identification, tracking and drift analysis* - A way of organizing the large amount of information retrieved from the web is categorizing the web pages into distinct topics. The categorization can be based on a similarity metric, which includes textual information and co-citation relations. Clustering or classification techniques can automatically and effectively identify relevant topics and add them in a topic-wise collection library.

   Adding a new document to a collection library includes:

   (i) Assigning each document to an existing topic (category)

   (ii) Re-checking of collection for the emergence of new topics

   (iii) Tracking the number of views to a collection

   (iv) Identifying the drift in a topic(s)

2. *Concept hierarchy creation* - Concept hierarchy is an important tool for capturing the general relationship among web documents. Creation of concept hierarchies is important to understand a category and sub-categories to which a document belongs. The clustering algorithms leverage more than two clusters, which merge into a cluster. That is merging the sub• clusters into a cluster.

   Important factors for creation of concept hierarchy include:

   (i) Identifying the organization of categories, such as flat, tree or network

   (ii) Planning the maximum number of categories per document

   (iii) Building category dimensions, such as domain, location, time, application and privileges.

3. *Relevance of content* - Relevance or the applicability of web content can be measured with respect to any of the following basis:

   (i) Document relevance describes the usefulness of a given document in a specified situation.

   (ii) Query-based relevance is the most useful method to assess the relevance of web pages. Query-based relevance is used in information retrieval tools such as search engines. The method calculates the similarity between query (search) keywords and document. Similarity, results can be refined through additional information such as popularity metric as seen in Google or the term positions in AltaVista.

   (iii) User-based relevance is useful in personal aspects. User profiles are maintained, and similarity between the user profile and document is calculated. The relevance is often used in push notification services.

   (iv) Role/task-based relevance is quite similar to user-based relevance. Instead of a user, here the profile is based on a particular role or task. Multiple users can provide input to profile.

9.3.3 Web Usage Mining

Web usage mining discovers and analyses the patterns in click streams. Web usage mining also includes associated data generated and collected as a consequence of user interactions with web resources. Figure 9.7 shows three phases for web usage mining.

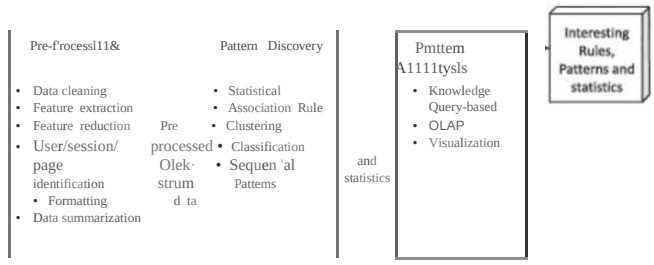


Figure 9.7 Process of web usage mining

The phases are:

1. Pre-processing - Converts the usage information collected from the various data sources into the data abstractions necessary for pattern discovery.
2. Pattern discovery - Exploits methods and algorithms developed from fields, such as statistics, data mining, ML and pattern recognition. 3. Pattern analysis - Filter outs uninteresting rules or patterns from the set found during the pattern discovery phase.

Usage data are collected at server, client and proxy levels. The usage data collected at the different sources represent the navigation patterns of the overall web traffic. This includes single-user, multi-user, single-site access and multi-site access patterns.

9.3.3.1 Pre-processing

The common data mining techniques apply on the results of pre-processing using vector space model (Refer Example 9.2).

Pre-processing is the data preparation task, which is required to identify:

(i) User through cookies, logins or URL information

(ii) Session of a single user using all the web pages of an application

(iii) Content from server logs to obtain state variables for each active session

(iv) Page references.

The subsequent phases of web usage mining are closely related to the smooth execution of data preparation task in pre-processing phase. The process deals with (i) extracting of the data, (ii) finding the accuracy of data, (iii) putting the data together from different sources, (iv) transforming the data into the required format and (iv) structure the data as per the input requirements of pattern discovery algorithm.

Pre-processing involves several steps, such as data cleaning, feature extraction, feature reduction, user identification, session identification, page identification, formatting and finally data summarization.

9.3.3.2 Pattern Discovery

The pre-processed data enable the application of knowledge extraction algorithms based on statistics, ML and data mining algorithms. Mining algorithms, such as path analysis, association rules, sequential patterns, clustering and classification enable effective processing of web usages. The choice of mining techniques depends on the requirement of the analyst. Pre-processed data of the web access logs transform into knowledge to uncover the potential patterns and are further provided to pattern analysis phase. Some of the techniques used for pattern discovery of web usage mining are:

Statistical techniques They are the most common methods which extract the knowledge about users. They perform different kinds of descriptive statistical analysis (frequency, mean, median) on variables such as page views, viewing time and length of path for navigational.

Statistical techniques enable discovering:

(i) The most frequently accessed pages

(ii) Average view time of a page or average length of a path through a site

(iii) Providing support for marketing decisions

Association rule The rules enable relating the pages, which are most often referenced together in a single server session. These pages may not be directly connected to one another using the hyperlinks.

Other uses of association rule mining are:

(i) Reveal a correlation between users who visited a page containing similar information. For example, a user visited a web page related to admission in an undergraduate course to those who search an eBook related to any subject.

(ii) Provide recommendations to purchase other products. For example, recommend to user who visited a web page related to a book on data analytics, the books on ML and Big Data analytics also.

(iii) Provide help to web designers to restructure their websites.

(iv) Retrieve the documents in prior in order to reduce the access time when loading a page from a remote site.

Clustering is the technique that groups together a set of items having similar features. Clustering can be used to:

(i) Establish groups of users showing similar browsing behaviors

(ii) Acquire customer sub-groups in e-commerce applications

(iii) Provide personalized web content to users

(iv) Discover groups of pages having related content. This information is valuable for search engines and web assistance providers.

Thus, user clusters and web-page clusters are two cases in the context of web usage mining. Web page clustering is obtained by grouping pages having similar content. User clustering is obtained by grouping users by their similarity in browsing behavior.

Model-based or distance-based clustering can be applied on web usage logs. The model type is often specified theoretically with model-based clustering. The model selection techniques and parameters estimate using maximum likelihood algorithms, such as Expectation Maximization (EM) determines the structure of model. Distance-based clustering measures the distance between pairs of web pages or users, and then groups the similar ones together into clusters. The most popular distance-based clustering techniques include partitional clustering and hierarchical clustering (Section 6.6.3).

Classification The method classifies data items into predefined classes. Classification is useful for:

(i) Developing a profile of users belonging to a particular class or category

(ii) Discovery of interesting rules from server logs. For example, 3750 users watched a certain movie, out of which 2000 are between age 18 to 23 and 1500 out of these lives in metro cities.

Classification can be done by using supervised inductive learning algorithms, such as decision tree classifiers, Naive Bayesian classifiers, k-nearest neighbour classifiers, support vector machines.

Sequential pattern discovery User navigation patterns in web usage data gather web page trails that are often visited by users in the order in which pages are visited. Markov Model can be used to model navigational activities in the website. Every page view in this model can be represented as a state. Transition probability between two states can represent the probability that a user will navigate from one state to the other. This representation allows for the computation of a number of significant user or site metrics that can lead to useful rules, pattern, or statistics.

9.3.3.3 Pattern Analysis

The objective of pattern analysis is to filter out uninteresting rules or patterns from the rules, patterns or statistics obtained in the pattern discovery phase.

The most common form of pattern analysis consists of:

(i) A knowledge query mechanism such as SQL

(ii) Another method is to load usage data into a data cube in order to perform Online Analytical Processing (OLAP) operations

(iii) Visualization techniques, such as graphing patterns or assigning the colors to different values, can often highlight overall patterns or trends in the data

(iv) Content and structure information can filter out patterns containing pages of a certain usage type, content type or pages that match a certain hyperlink structure.

Data cube enables visualizing data from different angles. For example, *toys* data visualization using category, colour and children preferences. Another example, news from category, such as sports, success stories, films or targeted readers (children, college students, etc).

Self-Assessment Exercise linked to LO 9.Z

1. Defineweb mining. Discussthe broad classifications of web mining and their applications.
2. List the tasks in pre-processing of web contents.
3. How are web-content mining tasks performed using machine learning algorithms?
4. How are topic identification, tracking and drift analysis done?
5. List and explain three phases of web-usage mining.
6. Highlight the techniques used for pattern discovery in web-usage mining giving an example of each.

9.41 PAGE RANK, STRUCTURE Of WEB AND ANALYZING A WEB GRAPH

Sections 9.2 and 9.3 described text data and web contents analysis. Hyperlinks links exist between the web contents. Link analysis finds the answers to the following:

Pag!!Ran king, ;a nalysi s ofweb-structu~ and discovering hubs, authori• ties and communities in wel>-structure

1. Can a linked (web) page rank them higher or lower?
2. Can the links be modeled as edges of graphs, structure of web as graph network, and applied the tools same as for graph analytics?
3. Can web graph mining method analyze and find a link sending spams?
4. Does a set of links correspond to a hub? Do the links correspond to an authority?
5. Does a linked page has higher authority compared to others?

Links analysis applies to domains of social networks and e-mail. The following sub-sections describe the applications of link analysis:

9.4.1 Page Rank Definition

The in-degree (visibility) of a link is the measure of number of in-links from other links. The out-degree (luminosity) of a link is number of other links to which that link points.

PageRank definition according to earlier approaches

Assume a web structure of hyperlinks. Each hyperlink in-links to a number of hyperlinks and out-links to a number of pages. A page commanding higher authority (rank) has greater number of in-degrees than out-degrees. Therefore, one measure of a page authority can be in-degrees with respect to out-degrees.

PageRank refers to the authority of the page measured in terms of number of times a link is sought after.

PageRank definition according to the new approach

Earlier approach of page ranking based on in-links and out-links does not capture the relative authority (importance) of the parents.

Page and co-authors (1998) defined a page ranking method,5 which considers the entire web in place of local neighbourhood of the pages and considers the relative authority of the parent links (over children).

9.4.2 Web Structure

Web structure models as directed-graphs network-organization. Vertex of the directed graph models an anchor. Let n = number of hyperlinks at the page **U.** Assume **u** is a vector with elements u_1, u_2, **...** un. Each page Pg **(u)** has anchors, called hyperlinks. Page Pg (v) consists of text document with m number of hyper links. v is a vector with elements vi, v_z, **...** v_m. The m is number of hyper links at Pg (v). A vertex u directs to another Page V. A page Pg (v) may have number of hyperlinks directed by out-edges to other page Pg **(w).** Consider the following hypotheses:

1. Text at the hyperlink represents the property of a vertex u that describes the destination V of the out-going edge. 2. A hyperlink in-between the pages represents the conferring of the authority.

Pages **U** and **U¢** hyperlinks u and ue out-linking to Page V. Let Page U has three hyperlinks parenting three Pages, V one, W two, X two, U' one, and Y two, respectively. Figure 9.8 shows a web structure consisting of pages and hyperlinks.

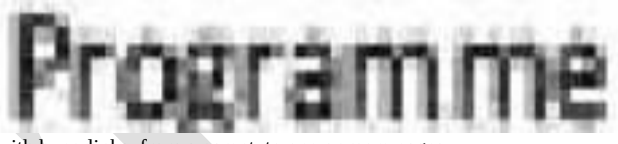


Figure 9.8 Web structure with hyperlinks from a parent to one or more pages

9.4.2.1 Dead Ends

Dead-end web pages refer to pages with no out-links. When a web page links to such pages, its page rank gets reduced. Dead ends are on a website having poor linking structure.

The web structure of service pages may have pages with a dead end. The end causes no further flows for further action and no internal links. Good website structures have the pages designed such that they specifically gently guide the visitors toward actions and towards next step. For example, if one searches for a book title on Amazon, then visitor gets links of other books also on a similar topic.

9.4.2.2 Analyzing and Implementing a System with Web Graph Mining

Number of metrics analyze a system using web graph mining. Following are the examples:

1. In-degrees and out-degrees
2. Closeness is centrality metric. Closeness, $Cc(v) = I/ \sum_{u} gdist(v,u)$, where gdist is the geodesic distance between vertex v with u and . V sum is over all u linked with V. Geodesic distance means the number of edges in a shortest path connecting two vertices. Assume v has an edge with w, and w has an edge with u. Assume u does not have direct edge from v. Then, geodesic distance= 2 (two edges between v and u in shortest path).
3. Betweenness

4. PageRank and LineRank

5. Hubs and authorities

6. Communities parameters, triangle count, clustering coefficient, K-neighbourhood

7. Top K-shortest paths

9.4.3 Computation of PageRank and PageRank Iteration

Assume that a web graph models the web pages. Page hyperlinks are the property of the graph node (vertex). Assume a Page, Pg (v) in-links from Pg (u), and Pg (u) out-linking similar to Pg (v), to total Nout [(Pg (u)] pages. Figure 9.9 shows Pg (v) in-links from Pg (u) and other pages.

Figure 9.9 Page Pg (v) in-links from Pg (u) and other pages

Nout for page U is 7 and for V is 1 in the figure. Number of in-linking Nin for page V is 4. Two algorithms to compute page rank are as follows:

1.PageRank algorithm using the in-degrees as conferring authority

Assume that the page U, when out-linking to Page V "considers" an equal fraction of its authority to all the pages it points to, such as Pgv. The following equation gives the initially suggested page rank, PR (based on in-degrees) of a page Pgv:

$$PR(Pgv) = DC \cdot \sum_{Pp:Pto \to P,.} [PR(Pga)/N(Pgu)) \quad (9.21)$$

where N(Pgu) is the total number of out-links from U. Sum is over all Pgv in-links. Normalization constant denotes by nc, such that PR of all pages sums equal to 1.

However, just measuring the in-degree does not account for the authority of the source of a link. Rank is flowing among the multiple sets of the links. When Pgv in-links to a page Pgu, its rank increases and when page Pgu out-links to other new links, it means that N (Pgu) increases, then rank PR(Pgv) sinks (decreases). Eventually, the PR (Pgv) converges to a value. Therefore, rank computation algorithm iterates the rank flowing computations as shown below:

EXAMPLE 9.7

Assume S corresponds to a set of pages. Initialize 'V Pg E S. Symbols mean that initialize all pages Pg contained in the S and initialize Page Rank (Pgv) for each page as follows:

$$PRinit(Pgv) = 1/|S| \tag{9.22}$$

How are the page ranks of the pages in a given set of pages iterated and computed till the ranks do not change (within specified margin, that means untill converge)?

SOLUTION

Iterate and compute PR (Pgv) for each page as follows:

Until ranks do not change (within specified margin) (that means converge)

for each Pgv E S compute,

$$PR(Pgv) = \sum_{P,u:Pgu \to \sim} [PR(PguJ/N(PguJ] \tag{9.22}$$

and normalization constant,

$$nc = \sum_{Jl,YES} [PRnew(Pgv)] \tag{9.23a}$$

$$\text{for each Pgv e S: } PR(Pgv) = nc\ PR(Pgv) \tag{9.23b}$$

2. PageRank algorithm using the relative authority of the parents over linked children

A method of PageRank considers the entire web in place of local neighbourhood of the pages and considers the relative authority of the parents (children). The algorithm uses the relative authority of the parents (children) and adds a rank for each page from a rank source.

The PageRank method considers assigning weight according to the rank of the parents. Page rank is proportional to the weight of the parent and inversely proportional to the out-links of the parent.

Assume that (i) Page v (Pgv) has in-links with parent Page u (Pgu) and other pages in set PA (v) of parent pages to v that means E PA(v), (ii) R(v) is PageRank of Pgv, (iii) R (u) is weight (importance/rank) of Pgu, and (iv) ch (u) is weight of child (out-links) of Pgu. Then the following equation gives PageRank R (v) of link v:

$$R(v) = \sum_{\bullet e\ PA(v)} [R(11)\sim ch(u)I] \tag{9.25}$$

where PA(v) is a set of links who are parents (in-links) of link v. Sum is over all parents of v. nc is normalization constant whose sum of weights is 1.

Assume that a rank source E exists that is addition to the rank of each page R (v) by a fixed rank value E(v) for Pgv. E(v) is fraction a of[l/lPA(v)I].

An alternative equation is as follows:

$$R(v) = nc\text{-}\{(1\text{-}a) \sum_{\bullet ePA(v)} [\,]_{1\sim(u)1} + a\text{-}E(v)\}. \quad (9.26)$$

where nc = [1/R(v)]. R(v) is iterated and computed for each parent in the set PA(v) till new value of R(v) does not change within the defined margin, say 0.001 in the succeeding iterations.

Significance of *a* PageRank can be seen as modeling a "random surfer" that starts on a random page and then at each point: E(v) models the probability that a random link jumps (surfs) and connect with out-link to Pgv. R(v) models the probability that the random link connects (surf) to Pgv at any given time. The addition of E(v) solves the problem of Pgv by chance out-linking to a link with dead end (no outgoing links).

Therefore, rank computation algorithm iterates the rank flowing computations as shown in Example 9.8.

EXAMPLE 9.8

Assume PA corresponds to a set of parent pages to a page v. Initialize V Pg E PA (v). Symbols mean that initialize all pages u contained in the set of parent pages of PA (v) and initialize Page Rank R(v) for each page as follows:

R(v) = [1/IPA(v)I]

How are the page ranks of the pages in a given set of pages iterated and computed till the ranks do not change (within specified margin, that means untill converges)?

SOLUTION

Iterate and compute R (v) for each page as follows:

Until ranks do not change that means converges (within specified margin, say 0.001)

for each v E PA (v) compute,

$$R\,v) = n \cdot \{(1\text{-}a \sum_{OE\ PA(v} [\text{—} \quad v)$$

and normalization constant

$$n = \sum_{uE\ P.\ v} [R\ v)$$

f r h v E PA v: R = n R v

PageRank Iteration using MapReduce functions in Spark Graph

The computation of PageRank using SparkGraph method (Section 8.5),

```
graph.pageRank(0.0001) .vertices ranksByUsername = users.join(ranks) .map{case id,
(username, rank)) => (username, rank).
```

The method includes conversions to MapReduce functions and using HDFS compatible files. Functions PageRank (), ranksByUsername () do the computations using the PageRankObject. GraphX consists of these functions (GraphOps). Graphx Operators includes the functions (Section 8.5).

Static PageRank algorithm runs for a fixed number of iterations, while dynamic PageRank runs until the computed rank converges. Convergence means that after certain iterations, the rank does not change significantly and any change remains within a pre• specified tolerance. Thereafter the iterations stop.

Assume specified tolerance at the start of iterations is 0.0001 (1 in 10000). When the rank does not change beyond that tolerance, it means rank value will converge and then the iterative process will stop.

9.4.4 Topic Sensitive PageRank and Link Spam

Number of methods have been suggested for computations of topic-sensitive page ranking, RTs· The RTs (v) of a page P (v) may be higher for a specific topic compared to other topics. A topic associates with a distinct bag of words for which the page has higher probability of surfing than other bags for that topic.

Topic-sensitive PageRank method uses surfing weights (probabilities) for the pages containing the topic or bag of words corresponding to a topic. Method for creating topic-sensitive PageRank is to compute the bias to rank R(v) and thus increase the effect of certain pages containing that topic or bag of words.

Refer equation (9.25) for computations of R (v), and equation (9.26) for computations after introducing additional influence to the page. A method of introducing biasing is simple. It assumes that a rank source E exists that is additional having in-links from other pages, and thus adds to the rank of each page R (v) by a fixed (uniform) or non-uniform weight factor a. The factor a is a multiplication factor to actual in-links without the bias.

Recapitulate equation (9.26). Probability of random jump to page v is E(v). An alternative equation for topic sensitive PageRank, R(v) computation for page P (v) is as follows:

$$R(v) = n.:-\{o-a,)-P(v) \; \mathrm{l} \; [1\sim(U)j]+a, \; .E(v)\}.$$

Probability of random jump to page v is *E* (v). at = O for page unrelated to a topic a is not O for page related to a topic. at = surfing probability for in-links for a topic t. Further, coefficient (1- a) is considered as biasing factor depending on the web page P (v) selected for a queried topic *t.*

The page is having in-links from other pages. Assume N, is number of topics to which a page is sensitive to surfing those topics. Effect of topics on PageRanks increases by using a non-uniform N, x 1 personalization vector for surfing probability p. Higher a means higher p.

Assume that the topics are tl, t2, ... , tn. Fix the number Nt. RTs is to be computed for each of them. Therefore, compute for each topic tj' the PageRank scores of page v as a function of tj, which means compute R (v, j), where j = 1, 2, ... , n. That also means compute the n elements of a non-uniform N, x 1 personalization vector for Rrs (v) for tl, t2, ... , tn.

Link Spam

Effects of a *link spam* can be nullified using the topic-sensitive PageRank algorithm. Link Spam tries to mislead the PageRank algorithm. A link spam attempts to make PageRank algorithm ineffective. The spam assisting pages connects to the page repeatedly and increases the in-degree of a page, thereby enhancing the rank to a large value.

A link spam creator website ws also has a page *ls* for whom *ws* attempts to enhance the PageRank. The w_5 has a large number of assisting pages al_5 which out-links to *ls* only. The *als* pages also prevent the PageRank of l_5 from being lost. A spam mass consists of w_5, *ls* and its *als* pages.

Methods nullify the effect by introducing a trust rank for a page u used in equation (9.29) and tracing spam mass of in-link pages to the page v.

Following are the steps for finding spam mass:

1. A distant topic sensitive page has unusually high in-degrees compared to the other pages of the same topic. A plot known as power-law plot is drawn between the log of number of web pages on the y-axis out-linking to the page v and logs of them in• degrees of v on the x-axis.
2. Plot is nearly linear as the number exponential decays is within degrees. N is proportional to exp (-d), where d is decay constant.
3. An unusual pattern with marked deviation from near linearity identifies the distant link spam mass.

9.4.5 Hubs andAuthorities

A hub is an index page that out-links to a number of content pages. A content page is topic authority. An authority is a page that has recognition due to its useful, reliable and significant information.

Figure 9.lO(a) shows hubs (shaded circles) with the number of out-links associated with each hub. Figure 9.IO(b) shows authorities (dotted circles) with the number of in-links and out-links associated with each link.

LO 1.3

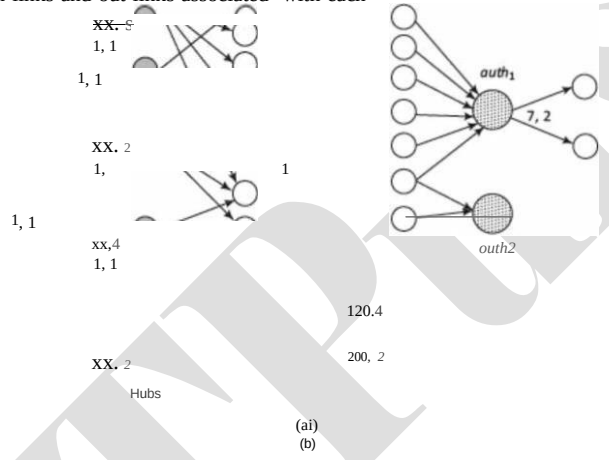


Figure 9.10 (a) Hubs (shaded circles) and (b) Authorities (dotted circles)

In-degrees (number of in-edges from other vertices) can be one of the measures for the authority. However, in-degrees do not distinguish between an in-link from a greater authority or lesser authority.

Authority, $auth_1$ in Figure 9.lO(b) has in-links from 6 vertices (in-degrees = 6) and $auth_2$ has in-links to just 2 (in-degree = 2). However, $auth_1$ has link with six vertices with in-degrees = 1, 1, 1, 1, 1 and 120 (total = 125). Authority, $auth_2$ has links with two vertices with in-degrees= 120 and 200 (total= 220). $Auth_2$has association with greater authorities. Therefore, in-degrees may not be a good measure as compared to authority.

Kleinberg (1998) developed the Hypertext-Induced Topic Selection (HITS) algorithm.6 The algorithm computes the hubs and authorities on a specific topic t. The HITS analyses a sub-graph of web, which is relevant to t. Basis of computation is (i) hubs are the ones, which out-link to number of authorities, and
(ii) authorities are the ones, which in-link to number of hubs. A bipartite graph exists for the hubs and authorities.

Consider a specifically queried topic t. Following are the steps:

1. Let a set of pages discover a root set R using standard search engine. Root pages may limit to top 200 for *t.*
2. Find a sub-graph of pages S, using a query that provides relevant pages for t and pointed by pages at R. Sub-graph S pages form Set for computations as it includes the children of parent R and limit to a random set of maximum 50 pages returned by a "reverse link" query.
3. Eliminate purely navigational links and links between two pages on the same host.
4. Consider only u (llull = 4-8) pages from a given hyperlink as pointer to any individual page. (Section 9.4.2)

Sub-graph for HITS consisting of root set R of pages and children of parents in the sub-graph S. Figure 9.11 shows subgraph S for HITS consisting of root set R of pages and all the pages pointed to by any page of R .

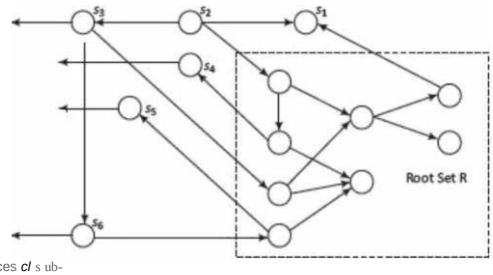


Figure 9.11 Sub-graph for HITS consisting of root set R of pages and base sub-graph *S* including all the pages pointed to by any page of *R.*

The left directed leftmost arrows from s3, s4 , s5 and s6 are pointing to nodes in sub-graph(s) associated to S. The following example explains the algorithm steps to compute hub score and authority score.

EXAMPLE 9.9

Assume that v has number of in-links and v has number of out-links. Assume S corresponds to base set of pages and R corresponds to root set. (i) Initialize S to R. (ii) Initialize '<:/ u E S. Symbols mean that initialize all pages u, contained in the S. (iii) Normalization constant is nc. The (i) hub (v) hub score and (ii) auth authority score of page v for each page is as follows:

For each v E S, auth (v) = 1; hub(v) = 1; nc = 1; (9.30)

How are the hub and authority of pages in a given set of pages iterated and computed till the ranks do not change (within specified margin, that means untill converges)? Usually 20 iterations converge the result within margin, usually set to 0.001.

SOLUTION

Iterate and compute auth (v) and hub (v) for each page as follows:

Until ranks do not change (within specified margin) (that means converges)

for each v E S compute,

$$\text{authrvj}_{e}\ \text{nc-}\ L\ [\text{hub(ul]}. \qquad (9.\text{Jla})$$

$$\text{hub(v)=nc2}\cdot\ L\ [\text{auth(u)}]. \qquad (9.\text{Jlb})$$

u:v~a and normalization constant,

$$\text{nd} = \underset{\text{P,VES}}{L}\ \text{ll[auth(v)f.} \qquad (9.32.1)$$

$$\text{nc2} = \underset{\text{P,VES}}{L}\ \text{ll[authMf.} \qquad (9.\text{J2b})$$

for each v e ~: authtv) = nc Lauthtv]: authtv) = nc l .autht v):

authtv): (9.J2c)

Difference between HITS and PageRank

HITS considers mutual reinforcement between authority and hub pages. PageRank ranks the pages just by authority and does not take into account distinctions between hubs and authorities. HITS considers the local neighbourhood between 4 to 8 pages surrounding the results of a query, whereas PageRank is applied to the entire web. HITS depends on topic t, while PageRank is topic• independent. PageRank effects by dead-ends.

9.4.6 Web Communities

Web communities are web sites or collections of websites, which limit the contents view and links to members. Examples of web communities are social networks, such as Linkedln, SlideShare, Twitter and Facebook.

The communities consist of sites for do-it-yourself sites, social networks, blogs or bulletin boards. The issues are privacy and reliability of information.

Metric for analysis of web-community sites are web graph parameters, such as triangle count, clustering coefficient and K• neighbourhood.

K-neighbourhood analysis means the number of 1st neighbour nodes, 2nd neighbour nodes, and so on (K = 1, 2, 3, 4 and so on).

K-core analysis means the number of cores within a marked area. A core may consist of a triangle of connected vertices. A core may consist of a rectangle with interconnected edges and diagonals. A core may also be a group of cores.

Spark Graphx (Section 8.5) described functions for degree centralities, degree distribution, separation of degree, betweenness centralities, closeness centralities, neighbourhoods, strongly connected components, triangle counts, PageRank, shortest path, Breadth First Search (BFS), minimum spanning tree (forest), spectral clustering and cluster coefficient.

9.4.7 Limitations of Link, Rank and Web Graph Analysis

Following are the limitations oflink and web graph analysis:

1. Search engines rely on metatags or metadata of the documents. That enhances the rank if metadata has biased information.
2. Search engines themselves may introduce bias while ranking the pages of clients higher as the pages of advertising companies may provide higher searches and hence lead to biased ranks.
3. A top authority may be a hub of pages on a different topic resulting in increased rank of the authority page.
4. Topic drift and content evolution can affect the rank. Off-topic pages may return the authorities.
5. Mutually reinforcing affiliates or affiliated pages/sites can enhance each other's rank and authorities.
6. The ranks may be unstable as adding additional nodes may have greater influence in rank changes.

Self-Assessment Exercise linked to LO 9.3

1. Write and explain the equations for computing PageRank using relative authority of parent nodes.
2. Show diagramatically network organization model of directed graphs for the structure of the web. How are the page hub and page authority computed?
3. What are the metrics which Spark GraphX compute?

4. How does the equation for computing the hub of a page differ from the computing authority of a page?
5. How does link spam function? How is the link spam discovered from the plot between the number of web pages and in• degrees?

9.5.1 Social Network as Graphs

9.51 SOCIAL NETWORKS AS GRAPHS AND SOCIAL NETWORK ANALYTICS

Representation of social networksas graphs, methods of social network analysis, finding *the* clustering in social netviork graphs, evaluating the Sim Ran counting triangles (cliques) and discovering the communities

A social network is a social structure made of individuals (or organizations) called "nodes," which are tied (connected) by one or more specific types of inter-dependency, such as friendship, kinship, financial exchange, dislike or relationships of beliefs, knowledge or prestige. *(Wikipedia)*

Social networking is the grouping of individuals into specific groups, like small rural communities or some other neighbourhoods based on a requirement. The following subsections describe social networks as graph, uses, characteristics and metrics.

Social network as graphs provide a number of metrics for analysis. The metrics enable the application of the graphs in a number of fields. Network topological analysis tools compute the degree, closeness, betweenness, egonet, K-neighbourhood, top-K shortest paths, PageRank, clustering, SimRank, connected components, K-cores, triangle count, graph matches and clustering coefficient. Bipartite weighted graph matching does collaborative filtering.

Apache Spark Graphx and IBM System G Graph Analytics tools are the tools for social network analysis.

Centralities, Ranking and Anomaly Detection

Important metrics are degree (centrality), closeness (centrality), betweenness (centrality) and eigenvector (centrality). Eigenvector consists of elements such as status, rank and other properties. Social graph-network analytics discovers the degree of interactions, closeness, betweenness, ranks, probabilities, beliefs and potentials.

Social network analysis of closeness and sparseness enables detection of abnormality in persons. Abnormality is found from properties of vertices and edges in network graph. Analysis enables summarization and find attributes for *anomaly*.

Social network characteristics from observations in the organizations are as follows:

1. Three-step neighbourhoods show positive correlation between a person and high performance. Betweenness between vertices and bridges between numbers of structures are not helpful to the organization. Too many strong links of a person may have a negative correlation with the performance.
2. Social network of a person shows high performance outcome when the network exhibits structural diversity. Person with a social network with an abundant number of structural holes exhibits higher performance. This is because having diverse relations help an organization.