MOHAMED SATHAK A.J. COLLEGE OF ENGINEERING

DEPARTMENT OF INFORMATION TECHNOLOGY

CS8091-BIG DATA ANALYTICS

Prepared by

Saranya.A

AP/CSE

UNIT I INTRODUCTION TO BIG DATA

Evolution of Big data – Best Practices for Big data Analytics – Big data characteristics – Validating – The Promotion of the Value of Big Data – Big Data Use Cases- Characteristics of Big Data Applications – Perception and Quantification of Value -Understanding Big Data Storage – A General Overview of High-Performance Architecture – HDFS – Map Reduce and YARN – Map Reduce Programming Model

UNIT II CLUSTERING AND CLASSIFICATION CS8091 Syllabus Big Data Analytics

Advanced Analytical Theory and Methods: Overview of Clustering – K-means – Use Cases – Overview of the Method – Determining the Number of Clusters – Diagnostics – Reasons to Choose and Cautions .- Classification: Decision Trees – Overview of a Decision Tree – The General Algorithm – Decision Tree Algorithms – Evaluating a Decision Tree – Decision Trees in R – Naïve Bayes – Bayes‘ Theorem – Naïve Bayes Classifier.

UNIT III ASSOCIATION AND RECOMMENDATION SYSTEM CS8091 Syllabus Big Data Analytics

Advanced Analytical Theory and Methods: Association Rules – Overview – Apriority Algorithm – Evaluation of Candidate Rules – Applications of Association Rules – Finding Association& finding similarity – Recommendation System: Collaborative Recommendation- Content Based Recommendation – Knowledge Based Recommendation- Hybrid Recommendation Approaches.

UNIT IV STREAM MEMORY 9 CS8091 Syllabus Big Data Analytics

Introduction to Streams Concepts – Stream Data Model and Architecture – Stream Computing, Sampling Data in a Stream – Filtering Streams – Counting Distinct Elements in a Stream – Estimating moments – Counting oneness in a Window – Decaying Window – Real time Analytics Platform(RTAP) applications – Case Studies – Real Time Sentiment Analysis, Stock Market Predictions. Using Graph Analytics for Big Data: Graph Analytics

UNIT V NOSQL DATA MANAGEMENT FOR BIG DATA AND VISUALIZATION CS8091 Syllabus Big Data Analytics

NoSQL Databases : Schema-less Models‖: Increasing Flexibility for Data Manipulation-Key Value Stores- Document Stores – Tabular Stores – Object Data Stores – Graph Databases Hive – Shading –- Hbase – Analyzing big data with twitter – Big data for E-Commerce Big data for blogs – Review of Basic Data Analytic Methods using R.

INTRODUCTION

Big data is high-volume, high-velocity and high-variety information assets that demand cost- effective, innovative forms of information processing for enhanced insight and decision making.

Big data is a term applied to data sets whose size or type is beyond the ability of traditional relational databases to capture, manage and process the data with low latency.

Big data is larger, more complex data sets, especially from new data sources. These data sets are so voluminous that traditional data processing software just can't manage them.

SOURCES OF BIG DATA

This data comes from a wide variety of sources: sensorsused to gather climate information, posts to social media sites, digital pictures and videos, purchase transaction records and cell phone GPS signals, to name afew.

Artificial intelligence (AI), Mobile, Social Media and the Internet of Things (IoT) are driving data complexity through new forms and sources of data.

For example, big data comes from Sensors, Devices, Video/Audio, Networks, Log files, Transactional applications, Web, and Social media — much of it generated in real time and at a very large scale.

APPLICATIONS OF BIG DATA

Healthcare
- Predictive Analysis
- Tele-medicine
- Fitnesswearables
- Remote Monitoring

• Banking
- ATM's
- E-Wallets

• StockMarket
• Education /Academics
• Manufacturing
• E-Commerce
• IT
• Government
• SocialMedia

TYPES OF BIG DATA

1. StructuredData

It owns a dedicated data model. It also has a well defined structure, it follows a consistent order and it is designed in such a way that it can be easily accessed and used by person or a computer. Structured data is usually stored in well defined columns and databases.

– StructuredSchema
– Tables with rows and columns ofdata
– Example :DBMS,RDBMS

2. Semi-StructuredData

It is considered as another form of structured data. It inherits few properties of structured data, but major parts of this kind of data failures to have a definitive structure and also it does not obey the formal structure of data models such asRDBMS.

– Schema is not definedproperly
– JSON, XML,CSV,RSS
– Ex: Transactional history file, Logfile

3. UnstructuredData

Unstructured data is completely different of which neither has a structure nor obeys to follow formal structural rules of data models. It does not even have a consistent format and it found to be varying all the time. But rarely it has information related to data andtime.

– Heterogeneous Data
– Text file, Images, Videos,Audio.

BIG DATA CHARACTERISTICS

Big data characteristics are mere word that explain the remarkable potential of big data. In early stages development of big data and related terms there were only 3 V's (Volume, Variety, Velocity) considered as potential characteristics.

But ever growing technology and tools and variety of sources where information being received has potentially increased these 3 V's into 5 V's and still evolving.

The 5 V's are

- Volume
- Variety
- Velocity
- Veracity
- Value

olume

- Volume refers to the unimaginable amounts of information generated every second. This information comes from variety of sources like social media, cell phones, sensors, financial records, stock marketetc.

- Enterprises are awash with ever- growing data of all types, easily amassing terabytes and even peta bytes of information. Data has grown exponentially over the past few years than in the past few decades. Social media, web portals and real time data using sensors have increased the amount ofdata

- We are currently using distributed systems, to store data in several locations and brought together by a software Framework likeHadoop.

- For example, Facebook alone can generate about billion messages, 4.5 billion times that the "like" button is recorded, and over 350 million new posts are uploaded each day. Such a huge amount of data can only be handled by Big DataTechnologies.

variety

- Variety refers to the many types of data that are available. A reason for rapid growth of data volume is that the data is coming from different sources in variousformats.

- Big data extends beyond structured data to include unstructured data of all varieties: text, sensor data, audio, video, click streams, log files andmore.

  The variety of data is categorized as follows:
  - Structured –RDBMS
  - Semi Structured – XML, HTML, RDF,JSON
  - Unstructured- Text, audio, video, logs,images

velocity

- Velocity is the fast rate at which data is received and (perhaps) acted on. In other words itis the speed at which the data is generated and processed to meet the demands and challenges that lie in the path of growth anddevelopment.

- For time-sensitive processes such as catching fraud, big data must be analyzed as it streams into the enterprise to maximize its businessvalue.

- Climate control, Health care, Stock exchange predictions, Banking transactions and IoT generates billions of data everysecond.

veracity

- Data veracity, in general, is how accurate or truthful a data set may be. More specifically, when it comes to the accuracy of big data, it's not just the quality of the data itself but how trustworthy the data source, type, and processing of itis.

- The data quality of captured data can vary greatly, affecting the accurateanalysis.

value

- Value is the major issue that we need to concentrate on. It is not just the amount of data that we store or process. It is actually the amount of valuable, reliable and trustworthy data that needs to be stored, processed, analyzed to findinsights.

- Mine the data, i.e., a process to turn raw data into useful data. Value represents benefits of data to your business such as in finding out insights, results, etc. which were not possible earlier.

VALIDATING BIGDATA

- Validation is factors considered before Adopting Big Data Technology. Big data may helpful to an organization but without proper validation it may go wrong for organization benefits.

- In generaljust because big data is feasible within the organization, it does not necessarily mean that it isreasonable.

- Therefore clear value proposition for business improvements needs to be identified. Unless there are clear processes for determining the value proposition, there is a risk that it will remain a fad until it hits the disappointment phase of the hypecycle.

- As a way to properly ground any initiatives around big data, one initial task would be to evaluate the organization's fitness such as feasibility, reasonability, value, integrability, andsustainability.
- Validation done with some framework that determines a score for each of these factors rangingfrom 0 (lowest level) to 4 (highestlevel).

- The resulting scores can be reviewed. Each of these variables is, for the most part, somewhat subjective, but there are ways of introducing a degree of objectivity, especially when considering the value of bigdata.

THE PROMOTION OF THE VALUE OF BIG DATA

A thoughtful approach must differentiate between hype and reality, and one way to do this is to review the difference between what is being said about big data and what is being done with big data.

A scan of existing content on the "value of big data" sheds interesting light on what is being promoted as the expected result ofbig data analytics and, more interestingly, how familiar those expectations sound.

A good example is provided within an economic study on the value of big data undertaken and published by the Center for Economics and Business Research (CEBR) that speaks to the cumulative value of:

- optimized consumer spending as a result of improved targeted customermarketing;

- improvements to research and analytics within the manufacturing sectors to lead to new productdevelopment;

- improvements in strategizing and business planning leading to innovation and new start-upcompanies;

- predictive analytics for improving supply chain management to optimize stock

management, replenishment, andforecasting;
- improving the scope and accuracy of frauddetection.

Curiously, these are exactly the same types of benefits promoted by business intelligence and data warehouse tools vendors and system integrators for the past 15 to 20 years,namely:

- Better targeted customermarketing
- Improved productanalytics
- Improved businessplanning
- Improved supply chain management
- Improved analysis for fraud, waste, andabuse

BIG DATA USECASES

Big data techniques can be used to leverage the business benefits and by increasing the value of an organization. Big data has beneficial in many applications and in general the following are the common categories. It is derived from The Apache Software Foundation's Powered By Hadoop Web site.

- **Business intelligence**, querying, reporting, searching, including many implementationof searching, filtering, indexing, speeding up aggregation for reporting and for report generation, trend analysis, search optimization, and general informationretrieval.

- **Improved performance** for common data management operations, with the majority focusing on log storage, data storage and archiving, followed by sorting, running joins, extraction/transformation/ loading (ETL) processing, other types of data conversions, as well as duplicate analysis andelimination.

- **Non-database applications**, such as image processing, text processing in preparation for publishing, genome sequencing, protein sequencing and structure prediction, web crawling, and monitoring workflowprocesses.

- **Data mining and analytical applications**, including social network analysis, facial recognition, profile matching, other types of text analytics, web mining, machine learning, information extraction, personalization and recommendation analysis, ad optimization, and behavioranalysis.

In turn, the core capabilities that are implemented using the big data application can be further abstracted into more fundamental categories:

- **Counting functions** applied to large bodies of data that can be segmented and distributed among a pool of computing and storage resources, such as document indexing, concept filtering, and aggregation (counts andsums).

- **Scanning functions** that can be broken up into parallel threads, such as sorting, data transformations, semantic text analysis, pattern recognition, andsearching.

- **Modeling** capabilities for analysis andprediction.

- **Storing** large datasets while providing relatively rapidaccess.

Generally, Processing applications can combine these core capabilities in different ways.

CHARACTERISTICS OF BIG DATA APPLICATIONS

The big data approach is mostly suited to addressing or solving business problems that are subject to one or more of the followingcriteria**:**

1) Data throttling: The business challenge has an existing solutions, but on traditional hardware, the performance of a solution is throttled as a result of data accessibility, data latency, data availability, or limits on bandwidth in relation to the size of inputs.

2) Computation-restricted throttling: There are existing algorithms, but they are heuristic and have not been implemented because the expected computational performance has not been met with conventionalsystems.

3) Large data volumes: The analytical application combines a multitude of existing large datasets and data streams with high rates of data creation anddelivery.

4) Significant data variety: The 11data in the different sources vary in structure and content, and some (or much) of the data is unstructured.

5) Benefits from data parallelization: Because of the reduced data dependencies, the application's runtime can be improved through task or thread-level parallelization applied to independent datasegments.

These criteria can be used to assess the degree to which business problems are suited to big data technology.

PERCEPTION AND QUANTIFICATION OF VALUE

So far we have looked at two facets of the appropriateness of big data, with the first being organizational fitness and the second being suitability of the business challenge.

The third facet must also be folded into the equation, and that is big data's contribution to the organization.

In essence, these facets drill down into the question of value and whether using big data significantly contributes to adding value to the organization by:

A. Increasing revenues: As an example, an expectation of using a recommendation engine would be to increase same-customer sales by adding more items into the marketbasket.

B. Lowering costs: As an example, using a big data platform built on commodity hardware for ETL would reduce or eliminate the need for more specialized servers used for data staging, thereby reducing the storage footprint and reducing operatingcosts.

C. Increasing productivity: Increasing the speed for the pattern analysis and matching done for fraud analysis helps to identify more instances of suspicious behavior faster,allowing

for actions to be taken more quickly and transform the organization from being focused on recovery of funds to proactive prevention of fraud.

D. Reducing risk: Using a big data platform or collecting many thousands of streams of automated sensor data can provide full visibility into the current state of a power grid, in which unusual events could be rapidly investigated to determine if a risk of an imminent outage can bereduced.

UNDERSTANDING BIG DATA STORAGE

TWO APPROACHES CONSIDERED FOR BIG DATA
Some algorithms expect that massive amounts of data are immediately available quickly, necessitating large amounts of core memory.

Other applications may need numerous iterative exchanges of data between different computing nodes, which would require high-speed networks.

STORAGE CONSIDERATIONS
The need to acquire and manage massive amounts of data suggests a need for specialty storage systems to accommodate the big data applications.

When evaluating specialty storage offerings, some variables to consider include:

- Scalability- Whether storage subsystem can support massive data volumes of increasing size.
- Extensibility - how flexible the storage system's architecture is in allowing the system to be grown without the constraint of artificiallimits.
- Fault tolerance, which imbues the storage environment with the capability to recover from intermittentfailures.
- Integratability, which measures how well the storage environment can be integrated into the productionenvironment.
- Accessibility - limitations or constraints in providing simultaneous access to an expanding user community without compromisingperformance
- High-speed I/O capacity, which measures whether the input/output channels can satisfy the demanding timing requirements for absorbing, storing, and sharing large data volumes.

All big data applications achieve their performance and scalability through deployment on a collection of storage and computing resources.

The ability to design, develop, and implement a big data application is directly dependent on an awareness of the architecture of the underlying computing platform, It includes the following four resources,

1. Processing capability, often referred to as a CPU, processor, or node. Modern processing nodes often incorporate multiple cores that are individual CPUs that share the node's memory and are managed and scheduled together, allowing multiple tasks to be run simultaneously; this is known asmultithreading.

2. Memory, which holds the data that the processing node is currently working on. Most single node machines have a limit to the amount ofmemory.

3. Storage, providing persistence of data—the place where datasets are loaded, and from which the data is loaded into memory to beprocessed.

4. Network, which provides the "pipes" through which datasets are exchanged between different processing and storagenodes.

Because single-node computers are limited in their capacity, they cannot easily accommodate massive amounts of data.

That is why the high-performance platforms are composed of collections of computers in which the massive amounts of data and requirements for processing can be distributed among a pool of resources.

A GENERAL OVERVIEW OF HIGH-PERFORMANCE ARCHITECTURE

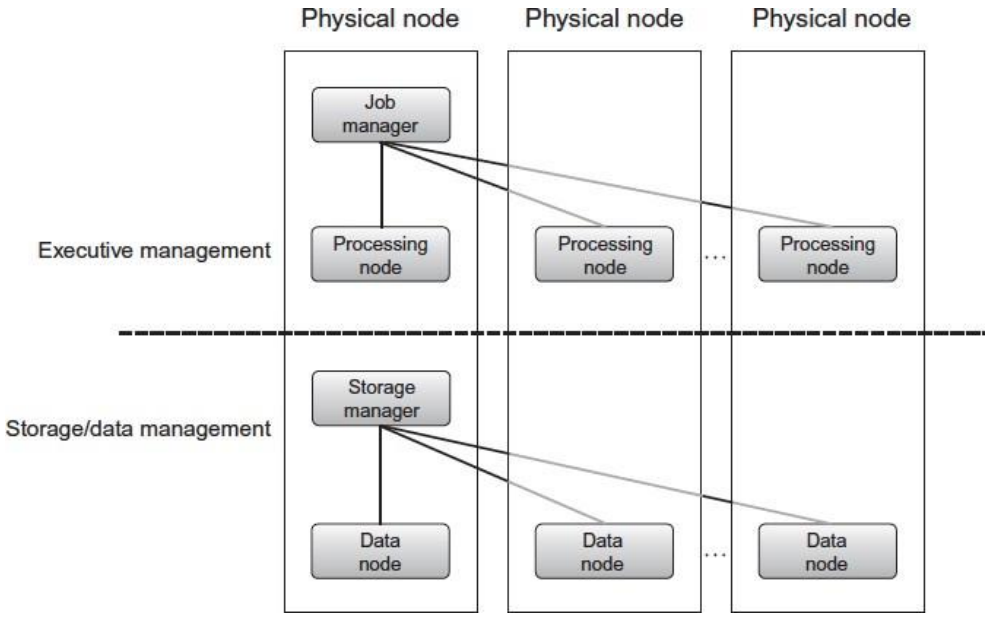
Most high-performance platforms are created by connecting multiple nodes together via a variety of network topologies.

The general architecture distinguishes the management of computing resources (and corresponding allocation of tasks) and the management of the data across the network of storage nodes. It is depicted in following image. It is generally called as master / slave architecture.

In this configuration, a master job manager oversees the pool of processing nodes, assigns tasks, and monitors the activity.

At the same time, a storage manager oversees the data storage pool and distributes datasets across the collection of storageresources

While there is no apriori requirement that there be any colocation of data and processing tasks, it is beneficial from a performance perspective to ensure that the threads process data that is local, or close to minimize the costs of data access latency.



HDFS

Hadoop Distributed File System (HDFS): A distributed file system that provides high throughput access to application data.

- The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on large clusters (thousands of computers) of small computer machines in a reliable, fault-tolerantmanner.

- Unlike other distributed systems, HDFS is highly fault tolerant and designed using low costhardware.

- HDFS holds very large amount of data and provides easier access. To store such huge data, the files split into blocks are stored across multiplemachines.

- These files are stored in redundant fashion to rescue the system from possible data losses in case of failure.

HDFS Architecture

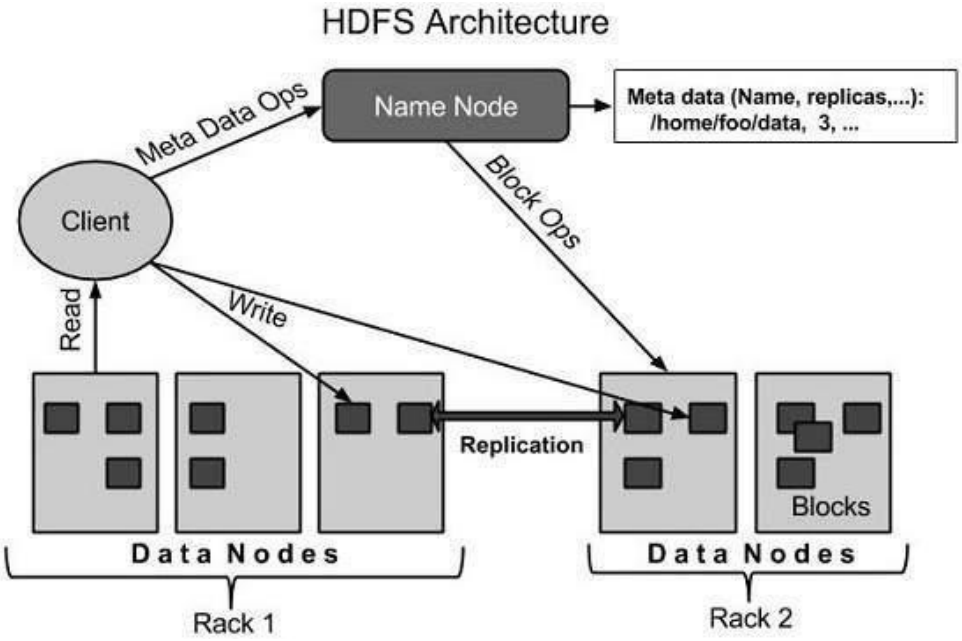
HDFS uses a master/slave architecture where master consists of a single Name Node that manages the file system metadata and one or more slave. Data Nodes that store the actual data.

Name Node

The name node is the commodity hardware that contains the GNU/Linux operating system and the name node software. It is a software that can be run on commodity hardware. The system having the name node acts as the master server and it does the following tasks:

- Manages the file systemnamespace.
- Stores metadata for the files, like the directory structure of a typical FileSystem.

- Regulates client's access tofiles.
- It also executes file system operations such as renaming, closing, and opening files and directories. It also determines the mapping of blocks to DataNodes.



Data Node:

The data node is a commodity hardware having the GNU/Linux operating system and data node software. These nodes manage the data storage of their system.

- A file in an HDFS namespace is split into several blocks and those blocks are stored in a set of DataNodes.
- Data nodes store and retrieve blocks when they are requested by client or namenode.
- They report back to name node periodically, with list of blocks that they arestoring.
- The data node also perform operations such as block creation, deletion and replication as stated by the name node.

Block

- Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual datanodes.
- These file segments are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called aBlock.
- The default block size is 64MB, but it can be increased as per the need to change in HDFSconfiguration.

Goals of HDFS

- Fault detection and recovery: Since HDFS includes a large number of commodity hardware, failure of components is frequent. Therefore HDFS should have mechanisms for quick and automatic fault detection andrecovery.
- Huge datasets: HDFS should have hundreds of nodes per cluster to manage the applications having hugedatasets.

- Hardware at data: A requested task can be done efficiently, when the computation takes place near the data. Especially where huge datasets are involved, it reduces the network traffic and increases thethroughput.

Features of HDFS

- It is suitable for the distributed storage andprocessing.
- Hadoop provides a command interface to interact withHDFS.
- The built-in servers of namenode and datanodehelp users to easily check the status of cluster.
- Streaming access to file systemdata.
- HDFS provides file permissions andauthentication.

Fault Tolerance

HDFS provides performance through distribution of data and fault tolerance through replication.

Failure management can be implemented in HDFS through

- Monitoring: There is a continuous "heartbeat" communication between the data nodes to the name node. If a data node's heartbeat isnot heard by the name node, the data node is considered to have failed and is no longer available. In this case, a replica is employed to replace the failed node, and a change is made to the replicationscheme.

- Rebalancing: This is a process of automatically migrating blocks of data from one data node to another when there is free space, when there is an increased demand for the data and moving it may improve performance (such as moving from a traditional disk drive to a solid-state drive that is much faster or can accommodate increased numbersof simultaneous accesses), or an increased need to replication in reaction to more frequent node failures.

- Managing integrity: HDFS uses checksums, which are effectively "digital signatures" associated with the actual data stored in a file(often calculated as a numerical function of the values within the bits of the files) that can be used to verify that the data stored corresponds to the data shared or received. When the checksum calculated for a retrieved block does not equal the stored checksum of that block, it is considered an integrity error. In that case, the requested block will need to be retrieved from a replicainstead.

- Metadata replication: The metadata files are also subject to failure, and HDFS can be configured to maintain replicas of the corresponding metadata files to protect against corruption.

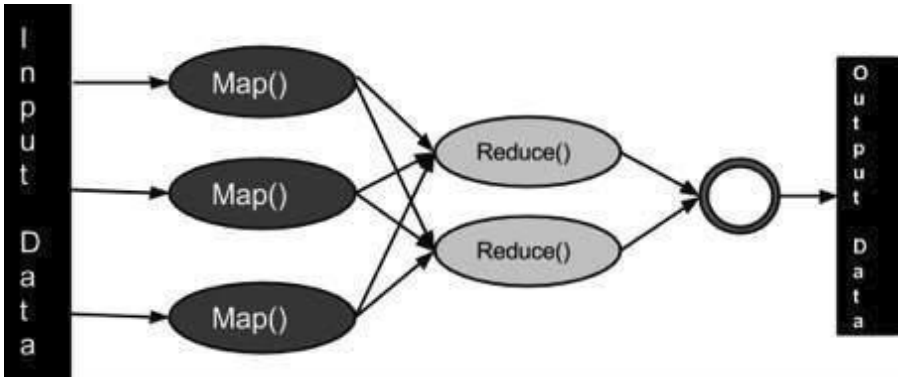
- Snapshots: This is incremental copying of data to establish a point in time to which the system can be rolledback.

MAPREDUCE

- Map Reduceis a programming model for writing applications that can process Big Data inparallel on multiple nodes. Map Reduce provides analytical capabilities for analyzing huge volumes of complexdata.
- Map Reduceis a processing technique and a program model for distributed computing based onjava.
- The Map Reduce algorithm contains two important tasks, namely Map andReduce.
- The Map task takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key-valuepairs).
- The Reduce task takes the output from the Map as an input and combines those data tuples (key-value pairs) into a smaller set oftuples.
- The major advantage of Map Reduceis that it is easy to scale data processing over multiple computing nodes. Under the Map Reducemodel, the data processing primitives are called mappers andreducers.

The Algorithm

- Map Reduce program executes in three stages, namely map stage, shuffle stage, and reducestage.
- Map stage : The map or mapper's job is to process the input data. Generally the input datais in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks ofdata.
- Reduce stage: This stage is the combination of the Shuffle stage and the Reduce stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in theHDFS.

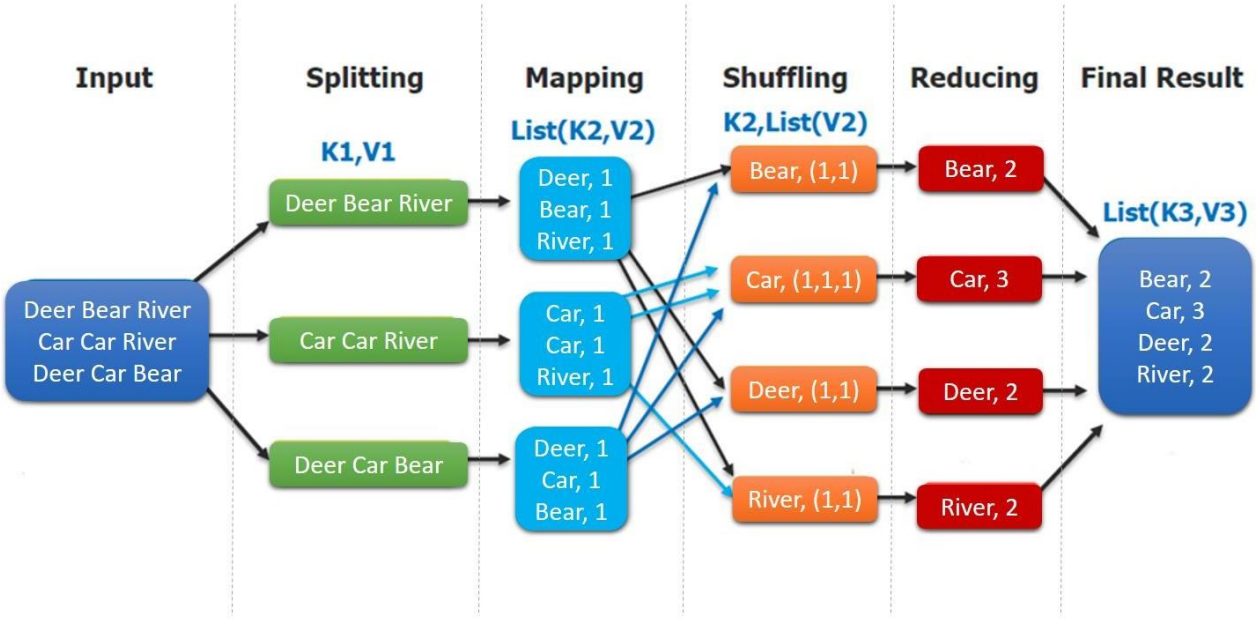


Phases

- Input Phase − Here we have a Record Reader that translates each record in an input file and sends the parsed data to the mapper in the form of key-valuepairs.
- Map − Map is a user-defined function, which takes a series of key-value pairs and processes each one of them to generate zero or more key-valuepairs.

- Intermediate Keys − They key-value pairs generated by the mapper are known as intermediate keys.

- Combiner − A combiner is a type of local Reducer that groups similar data from the map phase into identifiable sets. It takes the intermediate keys from the mapper as input and applies a user-defined code to aggregate the values in a small scope of one mapper. It is not a part of the main MapReduce algorithm; it isoptional.
- Shuffle and Sort − The Reducer task starts with the Shuffle and Sort step. It downloads the grouped key-value pairs onto the local machine, where the Reducer is running. The individual key-value pairs are sorted by key into a larger data list. The data list groups the equivalent keys together so that their values can be iterated easily in the Reducer task.
- Reducer − The Reducer takes the grouped key-value paired data as input and runs a Reducer function on each one of them. Here, the data can be aggregated, filtered, and combined in a number of ways, and it requires a wide range of processing. Once the execution is over, it gives zero or more key-value pairs to the finalstep.
- Output Phase − In the output phase, we have an output formatter that translates the final key-value pairs from the Reducer function and writes them onto a file using a record writer.



MapReduce Word Count Process

The MapReduce framework consists of a single master JobTracker and one slave TaskTracker per cluster-node.

- The master **Job Tracker** is responsible for resource management, tracking resource consumption/availability and scheduling the jobs component tasks on the slaves, monitoring them and re-executing the failed tasks.
- The slaves **TaskTracker**execute the tasks as directed by the master and provide task-status information to the masterperiodically.

There are three limitations of MapReduce
1.Network Latency
2.Does not support allapplications
3.ProcessorUnderutilization

YARN

- Hadoop Yet Another Resource Negotiator (YARN): This is a framework for job scheduling and cluster resource management. A resource management framework for scheduling and handling resource requests from distributedapplications.

- Issues in MapReduce are addressed in future versions of Hadoop through the segregation of duties within a revision calledYARN.

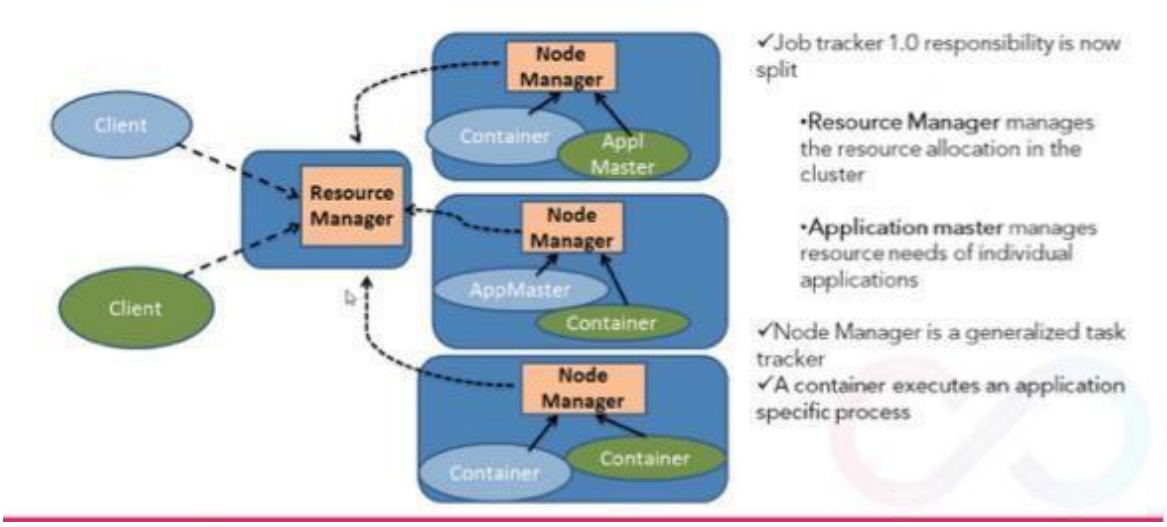
- In this approach, overall resource management has been centralized while management of resources at each node is now performed by a local NodeManager.

- In addition, there is the concept of an Application Master that is associated with each application that directly negotiates with the central Resource Manager for resources while taking over the responsibility for monitoring progress and trackingstatus.

- Pushing this responsibility to the application environment allows greater flexibility in the assignment of resources as well as be more effective in scheduling to improve nodeutilization.

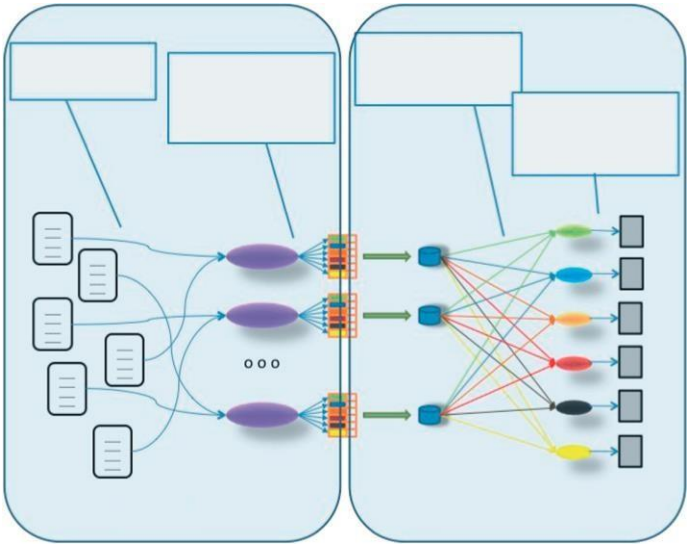
- Last, the YARN approach allows applications to be better aware of the data allocation across the topology of the resources within acluster.

- This awareness allows for improved colocation of compute and data resources, reducing data motion, and consequently, reducing delays associated with data access latencies. The result should be increased scalability andperformance.



THE MAPREDUCE PROGRAMMING MODEL

- Application development in MapReduce is a combination of the familiar procedural/imperative approaches used by Java or C++ programmers embedded within what is effectively a functional language programming model such as the one used within languages like Lisp andAPL.
- A MapReduce application is envisioned as a series of basic operations applied in a sequence to small sets of many (millions, billions, or even more) dataitems.
- These data items are logically organized in a way that enables the MapReduce execution model to allocate tasks that can be executed inparallel.
- The data items are indexed using a defined key into ,key, value. pairs, in which the key represents some grouping criterion associated with a computedvalue.
- Combining both data and computational independence means that both the data and the computations can be distributed across multiple storage and processing units and automatically parallelized.



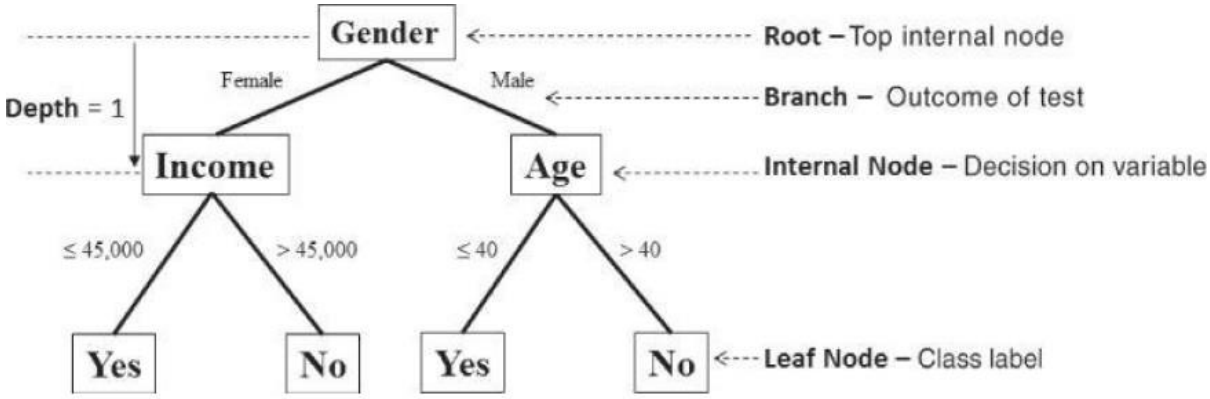
UNIT-II

DECISION TREE

- A decision tree (also called prediction tree) uses a tree structure to specify sequences of decisions andconsequences.
- Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, decision tree algorithm can be used for solving regression and classificationproblems.
- Given input X=(X1,X2..Xn), the goal is to predict a response or output variable Y. Each member of the set (X1,X2…Xn) is called an inputvariable.
- The prediction can be achieved by constructing a decision tree with test points and branches.
- At each test point, a decision is made to pick a specific branch and traverse down the tree. Eventually, a final point is reached, and a prediction can bemade.
- The input values of a decision tree can be categorical orcontinuous.
- A decision tree employs a structure of test points (called nodes) and branches, which represent the decision being made. A node without further branches is called a leaf node. The leaf nodes return classlabels.
- Decision trees have two varieties: classification trees and regressiontrees.
- Classification trees usually apply to output variables that are categorical—often binary—in nature, such as yes or no, purchase or not purchase, and soon.
- Regression trees, on the other hand, can apply to output variables that are numeric or continuous, such as the predicted price of a consumer good or the likelihood a subscription will bepurchased.

Overview of a DecisionTree

- The term branch refers to the outcome of a decision and is visualized as a line connecting two nodes.
- Internal nodes are the decision or test points. Each internal node refers to an input variable or an attribute. The top internal node is called the root. The branching of a node is referred to as asplit.

- The depth of a node is the minimum number of steps required to reach the node from the root.
- Leaf nodes are at the end of the last branches on the tree. They represent class labels—the outcome of all the prior decisions. The path from the root to a leaf node contains a series of decisions made at various internalnodes.
- 

The General Algorithm

The objective of a decision tree algorithm is to construct a tree T from a training set S.

If all the records in S belong to some class C, or if S is sufficiently pure (greater than a preset threshold), then that node is considered a leaf node and assigned the label C.

In contrast, if not all the records in S belong to class C or if S is not sufficiently pure, the algorithm selects the next most informative attribute A and partitions S according to A‘s values.

The algorithm constructs subtrees, T1,T2… for the subsets of S recursively until one of the following criteria is met:

– All the leaf nodes in the tree satisfy the minimum puritythreshold.

– The tree cannot be further split with the preset minimum puritythreshold.

– Any other stopping criterion is satisfied (such as the maximum depth of thetree).

The first step in constructing a decision tree is to choose the most informative attribute. A common way to identify the most informative attribute is to use entropy-basedmethods.

The entropy methods select the most informative attribute based on two basicmeasures:

– Entropy, which measures the impurity of anattribute

– Information gain, which measures the purity of anattribute

– Entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from thatinformation.

Given a class and its label x=X, let P(x) be the probability of x. Hx the entropy of x, is defined as

$$H_X = -\sum_{\forall x \in X} P(x) \log_2 P(x)$$

The next step is to identify the conditional entropy for each attribute. Given an attribute X, its value x, its outcome Y, and its value y, conditional entropy is the remaining entropy of given , formally defined as

$$H_{Y|X} = \sum_{x} P(x) H(Y|X = x)$$
$$= -\sum_{\forall x \in x} P(x) \sum_{\forall y \in Y} P(y|x) \log_2 P(y|x)$$

The information gain of an attribute A is defined as the difference between the base entropy and the conditional entropy of theattribute

$$InfoGain_A = H_S - H_{S|A}$$

Information gain compares the degree of purity of the parent node before a split with the degree of purity of the child node after asplit.

At each split, an attribute with the greatest information gain is considered the most informative attribute. Information gain indicates the purity of an attribute.

Decision Tree Algorithms

Multiple algorithms exist to implement decision trees, and the methods of tree construction vary with different algorithms.

Some popular algorithms include

• ID3,

• C4.5, and

• CART

ID3

ID3 (or Iterative Dichotomiser 3) is one of the first decision tree algorithms, and it was developed by John RossQuinlan.

Let A be a set of categorical input variables, P be the output variable (or the predicted class), and T be the training set.

ID3 follows the rule — A branch with an entropy of zero is a leaf node and A branch with entropy more than zero needs further splitting.

Algorithm

```
1 ID3 (A, P, T)
2 if T∈ϕ
3  return ϕ
4 if all records in T have the same value for P
5  return a single node with that value
6 if A∈ϕ
7  return a single node with the most frequent value of P in T
8 Compute information gain for each attribute in A relative to T
9 Pick attribute D with the largest gain
10 Let {d1,d2…dm} be the values of attribute D
11 Partition T into {T1,T2…Tm} according to the values of D
12 return a tree with root D and branches labeled d1,d2…dm
   going respectively to trees ID3(A-{D}, P, T1),
   ID3(A-{D}, P, T2), … ID3(A-{D}, P, Tm)
```

C4.5

- The C4.5 algorithm introduces a number of improvements over the original ID3 algorithm. The C4.5 algorithm can handle missingdata.
- If the training records contain unknown attribute values, the C4.5 evaluates the gain for an attribute by considering only the records where the attribute isdefined.
- Both categorical and continuous attributes are supported by C4.5. Values of a continuous variable are sorted andpartitioned.
- For the corresponding records of each partition, the gain is calculated, and the partition that maximizes the gain is chosen for the next split. The ID3 algorithm may construct a deep and complex tree, which would causeoverfitting.
- The C4.5 algorithm addresses the overfitting problem in ID3 by using a bottom-up technique called pruning to simplify the tree by removing the least visited nodes and branches.

CART

CART (or Classification And Regression Trees) is often used as a generic acronym for the decision tree, although it is a specific implementation.

Similar to C4.5, CART can handle continuous attributes. Whereas C4.5 uses entropy based criteria to rank tests, CART uses the Gini diversity index as defined below.

$$Gini_X = 1 - \sum_{\forall x \in X} P(x)^2$$

Whereas C4.5 employs stopping rules, CART constructs a sequence of subtrees, uses cross-validation to estimate the misclassification cost of each subtree, and chooses the one with the lowest cost.

Evaluating Decision Tree

- Decision trees use greedy algorithms, in that they always choose the option that seems the best available at thatmoment.
- At each step, the algorithm selects which attribute to use for splitting the remaining records. This selection may not be the best overall, but it is guaranteed to be the best at thatstep.
- This characteristic reinforces the efficiency of decision trees. However, once a bad split is taken, it is propagated through the rest of thetree.
- To address this problem, an ensemble technique (such as random forest) may randomize the splitting or even randomize data and come up with a multiple treestructure.
- These trees then vote for each class, and the class with the most votes is chosen as the predictedclass.

Evaluating

- There are a few ways to evaluate a decision tree. First, evaluate whether the splits of the tree make sense. Conduct sanity checks by validating the decision rules with domain experts, and determine if the decision rules aresound.
- Next, look at the depth and nodes of the tree. Having too many layers and obtaining nodes with few members might be signs of overfitting. In overfitting, the model fits the training set well, but it performs poorly on the new samples in the testingset.
- For decision tree learning, overfitting can be caused by either the lack of training data or the biased data in the trainingset.

Two approaches can help avoid overfitting in decision tree learning.

- Stop growing the tree early before it reaches the point where all the training data is perfectlyclassified.
- Grow the full tree, and then post-prune the tree with methods such as reduced-error pruning and rule-based postpruning.
- Last, many standard diagnostics tools that apply to classifiers can help evaluate overfitting.

Advantages of Decision Tree

- Decision trees are computationally inexpensive, and it is easy to classify thedata.
- Decision trees are able to handle both numerical and categorical attributes and are robust with redundant or correlatedvariables.
- Decision trees can handle categorical attributes with many distinct values, such as country codes for telephonenumbers.
- Decision trees can also handle variables that have a nonlinear effect on the outcome, so they work better than linear models (for example, linear regression and logistic regression) for highly nonlinearproblems.
- Decision trees can also be used to prune redundantvariables.

Disadvantages of Decision Tree

- Decision trees are not a good choice if the dataset contains many irrelevant variables. This is different from the notion that they are robust with redundant variables and correlatedvariables.
- Although decision trees are able to handle correlated variables, decision trees are not well suited when most of the variables in the training set are correlated, since overfitting is likely tooccur.

Naïve Bayes

- Naïve Bayes is a probabilistic classification method based on Bayes' theorem (or Bayes' law) with a fewtweaks.
- Bayes' theorem gives the relationship between the probabilities of two events and their conditional probabilities. Bayes' law is named after the English mathematician Thomas Bayes.
- A naïve Bayes classifier assumes that the presence or absence of a particular feature of a class is unrelated to the presence or absence of otherfeatures.
- The input variables are generally categorical, but variations of the algorithm can accept continuous variables. There are also ways to convert continuous variables into categorical ones. This process is often referred to as the discretization of continuousvariables.
- The output typically includes a class label and its corresponding probability score. The probability score is not the true probability of the class label, but it's proportional to the trueprobability.

Applications of Naïve Bayes Classifier

- Naïve Bayes classifiers are easy to implement and can execute efficiently even without prior knowledge of the data, they are among the most popular algorithms for classifying textdocuments.
- Spam filtering is a classic use case of naïve Bayes text classification. Bayesian spam filtering has become a popular mechanism to distinguish spam e-mail from legitimate e-mail.
- Naïve Bayes classifiers can also be used for frauddetection.
- In the domain of auto insurance, for example, based on a training set with attributes such as driver's rating, vehicle age, vehicle price, historical claims by the policy holder, police report status, and claim genuineness, naïve Bayes can provide probability-based classification of whether a new claim isgenuine.

The conditional probability of event C occurring, given that event A has already occurred, is denoted as P(C|A) , which can be found using the formula

$$P(C|A) = \frac{P(A \cap C)}{P(A)}$$

Above formula can be obtained with some minor algebra and substitution of the conditional probability:

$$P(C|A) = \frac{P(A|C) \cdot P(C)}{P(A)}$$

where C is theclasslabel $C \in \{c_1, c_2, \ldots c_n\}$ and

A is the observed attributes A={a1,a2,…am}. Second formula is the most common form of the Bayes' theorem.

Mathematically, Bayes' theorem gives the relationship between the probabilities of C and A, P(c) and P(A), and the conditional probabilities of C given A and A given C, namely P(C|A) and P(A|C).

Example 1

John flies frequently and likes to upgrade his seat to first class. He has determined that if he checks in for his flight at least two hours early, the probability that he will get an upgrade is 0.75; otherwise, the probability that he will get an upgrade is 0.35. With his busy schedule, he checks in at least two hours before his flight only 40% of the time. Suppose John did not receive an upgrade on his most recent attempt. What is the probability that he did not arrive two hours early?

Let C = {John arrived at least two hours early}, and A = {John received an upgrade}, then ¬C = {John did not arrive two hours early}, and ¬A = {John did not receive an upgrade}.

John checked in at least two hours early only 40% of the time, or P(C)=0.4 . Therefore,
$P(\neg C) = 1 - P(C) = 0.6$.

The probability that John received an upgrade given that he checked in early is 0.75, or P(A|C)=0.75.

The probability that John received an upgrade given that he did not arrive two hours early is 0.35, or

$P(A|\neg C) = 0.35$.

Therefore,

$P(\neg A|\neg C) = 0.65$.

The probability that John received an upgrade P(A) can be computed as shown

$$\begin{aligned} P(A) &= P(A \cap C) + P(A \cap \neg C) \\ &= P(C) \cdot P(A|C) + P(\neg C) \cdot P(A|\neg C) \\ &= 0.4 \times 0.75 + 0.6 \times 0.35 \\ &= 0.51 \end{aligned}$$

Thus, the probability that John did not receive an upgrade

$$P(\neg A) = 0.49$$

Using Bayes' theorem, the probability that John did not arrive two hours early given that he did not receive his upgrade is shown

$$\begin{aligned} P(\neg C|\neg A) &= \frac{P(\neg A|\neg C) \cdot P(\neg C)}{P(\neg A)} \\ &= \frac{0.65 \times 0.6}{0.49} \approx 0.796 \end{aligned}$$

Example 2

Assume that a patient named Mary took a lab test for a certain disease and the result came back positive. The test returns a positive result in 95% of the cases in which the disease is actually present, and it returns a positive result in 6% of the cases in which the disease is not present. Furthermore, 1% of the entire population has this disease. What is the probability that Mary actually has the disease, given that the test is positive?

Let C = {having the disease} and A = {testing positive}. The goal is to solve the probability of having the disease, given that Mary has a positive test result, P(C|A). From the problem description,

$P(C) = 0.01$, $P(\neg C) = 0.99$, $P(A|C) = 0.95$ and $P(A|\neg C) = 0.06$.

Bayes'theoremdefines $P(C|A) = P(A|C)P(C)/P(A)$

The probability of testing positive, that is P(A) , needs to be computed first. That computation is shown below

$$\begin{aligned} P(A) &= P(A \cap C) + P(A \cap \neg C) \\ &= P(C) \cdot P(A|C) + P(\neg C) \cdot P(A|\neg C) \\ &= 0.01 \times 0.95 + 0.99 \times 0.06 = 0.0689 \end{aligned}$$

According to Bayes' theorem, the probability of having the disease, given that Mary has a positive test result, is

$$P(C|A) = \frac{P(A|C)P(C)}{P(A)} = \frac{0.95 \times 0.01}{0.0689} \approx 0.1379$$

Naïve Bayes Classifier

With two simplifications, Bayes' theorem can be extended to become a naïve Bayes classifier.

The first simplification is to use the conditional independence assumption. That is, each attribute is conditionally independent of every other attribute given a class label Ci.

$$P(a_1, a_2, \ldots, a_m|c_i) = P(a_1|c_i)P(a_2|c_i)\cdots P(a_m|c_i) = \prod_{j=1}^{m} P(a_j|c_i)$$

Therefore, this naïve assumption simplifies the computation of

$P(a_1, a_2, \ldots, a_m|c_i)$.

The second simplification is to ignore the denominator. Because appears in the denominator of for all values of i, removing the denominator will have no impact on the relative probability scores and will simplify calculations.

- Naïve Bayes classification applies the two simplifications mentioned earlier and, as a result, $P(c_i|a_1, a_2, \ldots, a_m)$ is proportional to the product of $P(a_j|c_i)$ times $P(c_i)$.
- This is shown as below

$$P(c_i|A) \propto P(c_i) \cdot \prod_{j=1}^{m} P(a_j|c_i) \qquad i = 1, 2, \ldots n$$

- The mathematical symbol $\propto$ indicates that the LHS $P(c_i|A)$ is directly proportional to the RHS.

- Building a naïve Bayes classifier requires knowing certain statistics, all calculated from the training set.

- The first requirement is to collect the probabilities of all class labels, $P(c_i)$.

- The second thing the nai”ve Bayes classifier needs to know is the conditional probabilities of each attribute aj given each class label ci, namely *P(a* |cI

- Aftertrainingtheclassifierandcomputingalltherequiredstatistics,th e nai”veBayesclassifiercanbetestedoverthetestingset.

- Foreachrecordinthetestingset,thenai”veBayesclassifierassignst he classifier label Ci thatmaximizes

$$P(c_i)\cdot\prod_{j=1}^{m} P\left(a_j|c_i\right)$$

Because $P(c_i|a_1,a_2,\ldots,a_m)$ is proportional to the product of $P(a_j|c_i)(j\in[1,m])$ times (c_i) , the naïve BayPsc$0SSl(l€fdesigns the class liibel° n hlch results in the greiitestiulueover all . Thus, $P(c_i|a_1,a_2,\ldots,a_m)$ is

computed for each c_i with $\ldots,A)\propto P(c_i)\cdot\prod_{j=1}^{m} P\left(a_j|c_i\right)$.

UNIT 3

Association Rules

- Given a large collection of transactions in which each transaction consists of one or more items, association rules go through the items being purchased to see what items are frequently bought together and to discover a list of rules that describe the purchasing behavior.
- The goal with association rules is to discover interesting relationships among the items. Each of the uncovered rules is in the form X → Y, meaning that when item X is observed, item Y is alsoobserved.
- Using association rules, patterns can be discovered from the data that allow the association rule algorithms to disclose rules of related productpurchases.

- Association rules are sometimes referred to as ***market basket analysis***. Each transaction can be viewed as the shopping basket of a customer that contains one or more items. This is also known as anitemset.

- The term itemset refers to a collection of items or individual entities that contain some kind of relationship. This could be a set of retail items purchased together in one transaction, a set of hyperlinks clicked on by one user in a single session, or a set of tasks done in oneday.

- Given an itemset L, the support of L is the percentage of transactions that contain L. For example, if 80% of all transactions contain itemset {bread}, then the support of {bread} is 0.8. Similarly, if 60% of all transactions contain itemset {bread,butter}, then the support of {bread,butter} is0.6.

- A frequent itemset has items that appear together often enough. The term “often enough” is formally defined with a minimum support criterion. If the minimum support is set at 0.5, any itemset can be considered a frequent itemset if at least 50% of the transactions contain this itemset. In other words, the support of a frequent itemset should be greater than or equal to the minimumsupport.

- If an itemset is considered frequent, then any subset of the frequent itemset must also be frequent. This is referred to as the Apriori property (or downward closure property). For example, if 60% of the transactions contain {bread,jam}, then at least 60% of all the transactionswillcontain{bread}or{jam}.Inotherwords,whenthesupportof {bread,jam} is 0.6, the support of {bread} or {jam} is at least 0.6.

Apriori Algorithm

The Apriori algorithm takes a bottom-up iterative approach to uncovering the frequent itemsets by first determining all the possible items (or 1-itemsets, for example {bread}, {eggs}, {milk}, …) and then identifying which among them are frequent.

Assuming the minimum support threshold is set at 0.5, the algorithm identifies and retains those itemsets that appear in at least 50% of all transactions and discards (or "prunes away") the itemsets that have a support less than 0.5 or appear in fewer than 50% of the transactions.

In the next iteration of the Apriori algorithm, the identified frequent 1-itemsets are paired into 2-itemsets (for example, {bread,eggs}, {bread,milk}, {eggs,milk}, …) and again evaluated to identify the frequent 2-itemsets among them.

At each iteration, the algorithm checks whether the support criterion can be met; if it can, the algorithm grows the itemset, repeating the process until it runs out of support or until the itemsets reach a predefinedlength.

Let variable C_k be the set of candidate k-itemsets and variable be the set of k-itemsets that satisfy the minimum support. Given a transaction database , a minimum support threshold , and an optional parameter N indicating the maximum length an itemset could reach, Apriori iteratively computes frequent itemsets L_{k+1} based on L_k.

```
1 Apriori (D, δ, N)
2 k ← 1
3 L_k ← {1-itemsets that satisfy minimum support δ}
4 while L_k ≠ ∅
5  if ∄N ∨ (∃N ∧ k < N)
6  C_{k+1} ← candidate itemsets generated from L_k
7  for each transaction t in database D do
8   increment the counts of C_{k+1} contained in t
9  L_{k+1} ← candidates in C_{k+1} that satisfy minimum support δ
10  k ← k + 1
11 return ⋃_k L_k
```

The first step of the Apriori algorithm is to identify the frequent itemsets by starting with each item in the transactions that meets the predefined minimum support threshold .

These itemsets are 1-itemsets denoted as L_1, as each 1-itemset contains only one item. Next, the algorithm grows the itemsets by joining L_1 onto itself to form new, grown 2-itemsets denoted as L_2 and determines the support of each 2-itemset in L_2.

Those itemsets that do not meet the minimum support threshold are pruned away. The growing and pruning process is repeated until no itemsets meet the minimum support threshold.

Optionally, a threshold N can be set up to specify the maximum number of items the itemset can reach or the maximum number of iterations of the algorithm. Once completed, output of the Apriori algorithm is the collection of all the frequent k-itemset.

Next, a collection of candidate rules is formed based on the frequent itemsets uncovered in the iterative process described earlier. For example, a frequent itemset {milk,eggs} may suggest candidate rules {milk}→{eggs} and {eggs}→{milk}.

Evaluation of Candidate Rules

Confidence is defined as the measure of certainty or trustworthiness associated with each discovered rule. Mathematically, confidence is the percent of transactions that contain both X and Y out of all the transactions that containX.

$$Confidence(X \rightarrow Y) = \frac{Support(X \wedge Y)}{Support(X)}$$

For example, if {bread,eggs,milk} has a support of 0.15 and {bread,eggs} also has a support of 0.15, the confidence of rule {bread,eggs}→{milk} is 1, which means 100% of the time a customer buys bread and eggs, milk is bought as well. The rule is therefore correct for 100% of the transactions containing bread and eggs.

A relationship may be thought of as interesting when the algorithm identifies the relationship with a measure of confidence greater than or equal to a predefined threshold. This predefined threshold is called the minimum confidence.

Lift measures how many times more often X and Y occur together than expected if they are statistically independent of each other. Lift is a measure of how X and Y are really related rather than coincidentally happening together.

$$Lift(X \rightarrow Y) = \frac{Support(X \wedge Y)}{Support(X) * Support(Y)}$$

Leverage is a similar notion, but instead of using a ratio, leverage uses the difference. Leverage measures the difference in the probability of X and Y appearing together in the dataset compared to what would be expected if X and Y were statistically independent of each other.

$$Leverage(X \rightarrow Y) = Support(X \wedge Y) - Support(X) * Support(Y)$$

Confidence is able to identify trustworthy rules, but it cannot tell whether a rule is coincidental. A high-confidence rule can sometimes be misleading because confidence does not consider support of the itemset in the ruleconsequent.

Measures such as lift and leverage not only ensure interesting rules are identified but also filter out the coincidental rules.

Applications of Association Rules

The term market basket analysis refers to a specific implementation of association rules mining that many companies use for a variety of purposes, including these:

- Broad-scale approaches to better merchandising—what products should be included in or excluded from the inventory eachmonth
- Cross-merchandising between products and high-margin or high-ticketitems
- Physical or logical placement of product within related categories ofproducts
- Promotional programs—multiple product purchase incentives managed through a loyalty cardprogram

Besides market basket analysis, association rules are commonly used for recommender systems [and clickstream analysis.

Many online service providers such as Amazon and Netflix use recommender systems. Recommender systems can use association rules to discover related products or identify customers who have similar interests.

Recommendation System

There is an extensive class of Web applications that involve predicting user responses to options. Such a facility is called a recommendation system.

Two good examples of recommendation systems are:

1. Offering news articles to on-line newspaper readers, based on a prediction of readerinterests.

2. Offering customers of an on-line retailer suggestions about what they might like to buy, based on their past history of purchases and/or productsearches.

Types of Recommendation System

- Collaborativerecommendation
- Content-basedrecommendation
- Knowledge-basedrecommendation
- Hybridapproaches

Collaborative recommendation

Collaborative filtering systems recommend items based on similarity measures between users and/or items. The items recommended to a user are those preferred by similar users.

The basic idea of these systems is that if users shared the same interests in the past – if they viewed or bought the same books, for instance – they will also have similar tastes in the future.

The main idea of collaborative recommendation approaches is to exploit information about the past behavior or the opinions of an existing user community for predicting which items the current user of the system will most probably like or be interested in.

Pure collaborative approaches take a matrix of given user–item ratings as the only input and typically produce the following types of output:

(a) a (numerical) prediction indicating to what degree the current user will like or dislike a certain itemand

(b) a list of n recommendeditems.

Types of Collaborative recommendation

- User-based nearest neighborrecommendation
- Item-based nearest neighborrecommendation

User-based nearest neighbor recommendation

The main idea is simply as follows: given a ratings database and the ID of the current (active) user as an input, identify other users (sometimes referred to as peer users or nearest neighbors) that had similar preferences to those of the active user in the past.

Then, for every product p that the active user has not yet seen, a prediction is computed based on the ratings for p made by the peer users.

The underlying assumptions of such methods are that (a) if users had similar tastes in the past they will have similar tastes in the future and (b) user preferences remain stable and consistent over time.

We use U = {u1, . . . , un} to denote the set of users, P = {p1, . . . , pm} for the set of products (items), and R as an n × m matrix of ratings ri,j , with i∈1 . . . n, j ∈1 . . . m.

With respect to the determination of the set of similar users, one common measure used in recommender systems is ***Pearson's correlation coefficient***. The similarity sim(a, b) of users a and b, given the rating matrix R, is defined as

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \overline{r_a})(r_{b,p} - \overline{r_b})}{\sqrt{\sum_{p \in P}(r_{a,p} - \overline{r_a})^2}\sqrt{\sum_{p \in P}(r_{b,p} - \overline{r_b})^2}}$$

The Pearson correlation coefficient takes values from +1 (strong positive correlation) to −1 (strong negative correlation).

Item-based nearest neighbor recommendation

Large-scale e-commerce sites, often implement a different technique, item-based recommendation, which is more apt for offline preprocessing and thus allows for the computation of recommendations in real time even for a very large ratingmatrix.

The main idea of item-based algorithms is to compute predictions using the similarity between items and not the similarity between user.

The cosine similarity measure

To find similar items, a similarity measure must be defined. In item-based recommendation approaches, cosine similarity is established as the standard metric.

The similarity between two items a and b – viewed as the corresponding rating vectors a and b – is formally defined as follows:

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

The · symbol is the dot product of vectors. | a | is the Euclidian length of the vector, which is defined as the square root of the dot product of the vector with itself.

The basic cosine measure does not take the differences in the average rating behavior of the users into account. This problem is solved by using the adjusted cosine measure, which subtracts the user average from the ratings.

Let U be the set of users that rated both items a and b. The adjusted cosine measure is then calculated as follows:

$$sim(a, b) = \frac{\sum_{u \in U}(r_{u,a} - \overline{r_u})(r_{u,b} - \overline{r_u})}{\sqrt{\sum_{u \in U}(r_{u,a} - \overline{r_u})^2}\sqrt{\sum_{u \in U}(r_{u,b} - \overline{r_u})^2}}$$

Content-based recommendation

Content-Based systems focus on properties of items. Similarity of items is determined by measuring the similarity in theirproperties.

Content-based recommendation is based on the availability of item descriptions and a profile that assigns importance to thesecharacteristics.

In a content-based system, we must construct for each item a profile, which is a record or collection of records representing important characteristics of that item. In simple cases, the profile consists of some characteristics of the item that are easily discovered.

For example, consider the features of a movie that might be relevant to a recommendation system.

1. The set of actors of the movie. Some viewers prefer movies with their favoriteactors.

2. The director. Some viewers have a preference for the work of certaindirectors.

3. The year in which the movie was made. Some viewers prefer old movies; others watch only the latestreleases.

4. The genre or general type of movie. Some viewers like only comedies, others dramas or romances.

A typical similarity metric that is suitable for multivalued characteristics, we could, for example, rely on the Dice coefficient[2] as follows: If every book Bi is described by a set of keywords keywords(Bi), the Dice coefficient measures the similarity between books bi and bj as

$$\frac{2 \times |keywords(b_i) \cap keywords(b_j)|}{|keywords(b_i)| + |keywords(b_j)|}$$

The vector space model and TF-IDF

In a first, and very na¨ıve, approach, one could set up a list of all words that appear in all documents and describe each document by a Boolean vector, where a 1 indicates that a word appears in a document and a 0 that the word does not appear. Boolean vector approach has many shortcomings.

To solve the shortcomings of the simple Boolean approach, documents are typically described using the TF-IDF encoding format.

TF-IDF stands for term frequency-inverse document frequency. Text documents can be TF-IDF encoded as vectors in a multidimensional Euclidian space.

The space dimensions correspond to the keywords (also called terms or tokens) appearing in the documents. The coordinates of a given document in each dimension (i.e., for each term) are calculated as a product of two sub measures: term frequency and inverse document frequency.

Term frequency describes how often a certain term appears in a document (assuming that important words appear more often).

To take the document length into account and to prevent longer documents from getting a higher relevance weight, some normalization of the document length should be done.

We search for the normalized term frequency value TF(i, j) of keyword i in document j. Given a keyword i, let Other Keywords(i, j) denote the set of the other keywords appearing in j . ComputethemaximumfrequencymaxOthers(i,j)asmax(freq(z,j)),z∈Other Keywords(i,j
). Finally, calculate TF(i, j) as

$$TF(i, j) = \frac{freq(i, j)}{maxOthers(i, j)}$$

Inverse document frequency is the second measure that is combined with term frequency. It aims at reducing the weight of keywords that appear very often in all documents.

The inverse document frequency for i is typically calculated as

$$IDF(i) = log\frac{N}{n(i)}$$

The combined TF-IDF weight for a keyword i in document j is computed as the product of these two measures:

$$TF\text{-}IDF(i, j) = TF(i, j) * IDF(i)$$

In the TF-IDF model, the document is, therefore, represented not as a vector of Boolean values for each keyword but as a vector of the computed TF-IDF weights.

Similarity-based

retrieval Nearest

neighbors

Afirst, straightforward, method for estimating to what extent a certain document will be of interest to a user is simply to check whether the user liked similar documents in the past.

A k-nearest-neighbor method (kNN) has been implemented, for similarity based retrieval under nearest neighbors.

Relevance feedback – Rocchio's method

Another method that is based on the vector-space model and was developed in the context of the pioneering information retrieval (IR) system SMART is Rocchio's relevance feedback method.

Knowledge-based recommendation

A system that exploits additional and means–end knowledge to generate recommendations is called as knowledge based recommendation.

In such knowledge-based approaches, the recommender system typically makes use of additional, often manually provided, information about both the current user and the available items. Constraint based recommenders are one example of suchsystems.

The advantage of these systems is that no ramp-up problems exist, because no rating data are needed for the calculation of recommendations. Recommendations are calculated independently of individual user ratings: either in the form of similarities between customer requirements and items or on the basis of explicit recommendation rules.

Two basic types of knowledge-based recommender systems are

- Constraint basedand
- Case-basedsystems

Both approaches are similar in terms of the recommendation process: the user must specify the requirements, and the system tries to identify a solution. If no solution can be found, the user must change the requirements.

Case-based recommenders focus on the retrieval of similar items on the basis of different types of similarity measures, whereas constraint-based recommenders rely on an explicitly defined set of recommendationrules.

In constraint-based systems, the set of recommended items is determined by, for instance, searching for a set of items that fulfill the recommendation rules.

Case-based systems, on the other hand, use similarity metrics to retrieve items that are similar (within a predefined threshold) to the specified customer requirements.

Knowledge representation and reasoning

In general, knowledge-based systems rely on detailed knowledge about item characteristics.

A constraint-based recommendation problem can, in general, be represented as a constraint satisfaction problem that can be solved by a constraint solver or in the form of a conjunctive query that is executed and solved by a database engine.

Case-based recommendation systems mostly exploit similarity metrics for the retrieval of items from a catalog.

Constraints

A classical constraint satisfaction problem (CSP)1 can be described by a-tuple (V,D,C) where

V is a set of variables,

D is a set of finite domains for these variables, and

C is a set of constraints that describes the combinations of values the variables can simultaneously take.

A solution to a CSP corresponds to an assignment of a value to each variable in V in a way that all constraints are satisfied.

Constraint-based recommender systems can build on this formalism and exploit a recommender knowledge base that typically includes two different sets of variables (V = VC ∪ VPROD), one describing potential customer requirements and the other describing product properties.

Three different sets of constraint (C = CR ∪ CF ∪ CPROD) define which items should be recommended to a customer in which situation.

- Customer properties (VC) describe the possible customerrequirement.
- Product properties (VPROD) describe the properties of products in anassortment
- Compatibility constraints (CR) define allowed instantiations of customerproperties
- Filter conditions (CF) define under which conditions which products should beselected
- Product constraints (CPROD) define the currently available productassortment.

Cases and similarities

In case-based recommendation approaches, items are retrieved using similarity measures that describe to which extent item properties match some given user's requirements.

The so-called distance similarity of an item p to the requirements r ∈REQ is often defined as

In this context, sim(p, r) expresses for each item attribute value φr (p) its distance to the customer requirement r ∈REQ – for example, φmpix(p1) = 8.0. Furthermore, wr is the importance weight for requirement r.3

$$similarity(p, REQ) = \frac{\sum_{r \in REQ} w_r * sim(p, r)}{\sum_{r \in REQ} w_r}$$

In real-world scenarios, there are properties a customer would like to maximize – for example, the resolution of a digital camera. There are also properties that customers want to minimize – for example, the price of a digitalcamera.

In the first case we are talking about "moreis- better" (MIB) properties; in the second case the corresponding properties are denoted with "less-is-better" (LIB).

First, in the case of MIB properties, the local similarity between p and r is calculated as follows:

$$sim(p, r) = \frac{\phi_r(p) - min(r)}{max(r) - min(r)}$$

The local similarity between p and r in the case of LIB properties is calculated as follows:

$$sim(p, r) = \frac{max(r) - \phi_r(p)}{max(r) - min(r)}$$

Hybrid Recommendation System

Hybrid recommender systems are technical approaches that combine several algorithm implementations or recommendation components.

An excellent example for combining different recommendation algorithm variants is the Netflix Prize competition1, in which hundreds of students and researchers teamed up to improve a collaborative movie recommender by hybridizing hundreds of different collaborative filtering techniques and approaches to improve the overall accuracy.

Hybridization designs

There are three base designs: ***monolithic, parallelized, and pipelined hybrids***.

Monolithic denotes a hybridization design that incorporates aspects of several recommendation strategies in one algorithm implementation.

Several recommenders contribute virtually because the hybrid uses additional input data that are specific to another recommendation algorithm, or the input data are augmented by one technique and factually exploited by the other.

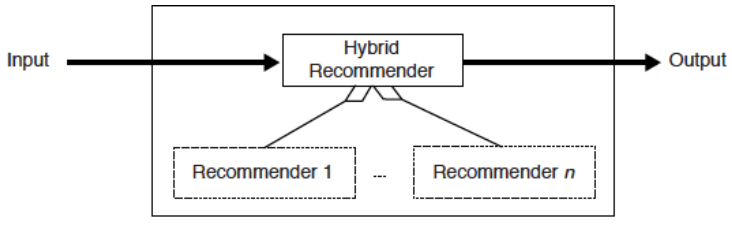


Figure 5.2. Monolithic hybridization design.

Parallelized hybrid recommender systems operate independently of one another and produce separate recommendation lists.

In a subsequent hybridization step, their output is combined into a final set of recommendations. Following Burke's taxonomy, the weighted, mixed, and switching strategies require recommendation components to work in parallel.

When several recommender systems are joined together in **a pipeline architecture**, the output of one recommender becomes part of the input of the subsequent one. Optionally, the subsequent recommender components may use parts of the original input data, too. The Cascade and meta-level hybrids, as defined by Burke, are examples of such pipelinearchitectures.

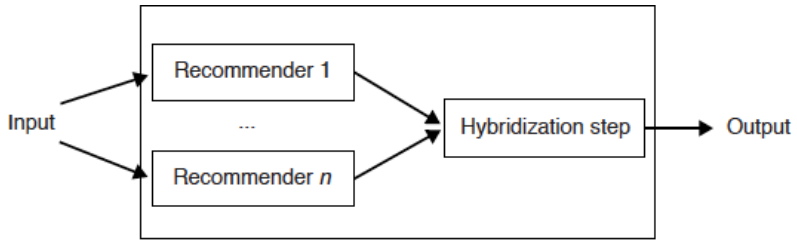


Figure 5.3. Parallelized hybridization design.

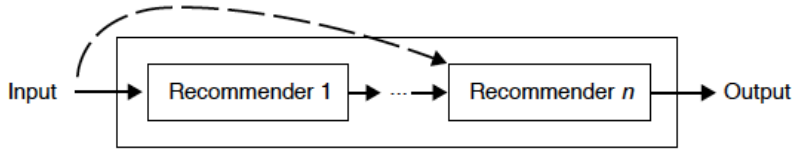


Figure 5.4. Pipelined hybridization design.

Monolithic hybridization design

Whereas the other two designs for hybrid recommender systems consist of two or more components whose results are combined, monolithic hybrids consist of a single recommender component that integrates multiple approaches by preprocessing and combining several knowledge sources.

Hybridization is thus achieved by a built-in modification of the algorithm behavior to exploit different types of input data. Typically, data-specific preprocessing steps are used to transform the input data into a representation that can be exploited by a specific algorithm paradigm.

Both feature combination and feature augmentation strategies can be assigned to this category.

A feature combination hybrid is a monolithic recommendation component that uses a diverse range of input data.

A feature combination hybrid that combines collaborative features, such as a user's likes and dislikes, with content features of catalog items.

Another approach for feature combination was proposed by Zanker and Jessenitschnig (2009b), who exploit different types of rating feedback based on their predictive accuracy and availability.

Feature augmentation is another monolithic hybridization design that may be used to integrate several recommendation algorithms. In contrast with feature combination, this hybrid does not simply combine and preprocess several types of input, but rather applies more complex transformation steps.

Parallelized hybridization design

Parallelized hybridization designs employ several recommenders side by side and employ a specific hybridization mechanism to aggregate their outputs.

Burke (2002b) elaborates on the mixed, weighted, and switching strategies. However, additional combination strategies for multiple recommendation lists, such as majority voting schemes, may also be applicable.

Mixed hybrids

A mixed hybridization strategy combines the results of different recommender systems at the level of the user interface, in which results from different techniques are presented together.

Therefore the recommendation result for user u and item i of a mixed hybrid strategy is the set of -tuples □score, k□ for each of its n constituting recommenders reck :

$$rec_{mixed}(u,i) = \bigcup_{k=1}^{n} \langle rec_k(u,i), k \rangle$$

Weighted hybrids

A weighted hybridization strategy combines the recommendations of two or more recommendation systems by computing weighted sums of theirscores.

Thus, given n different recommendation functions rec_k with associated relative weights β_k:

$$rec_{weighted}(u,i) = \sum_{k=1}^{n} \beta_k \times rec_k(u,i)$$

where item scores need to be restricted to the same range for all recommenders

and

$\sum_{k=1}^{n} \beta_k = 1$.

Obviously, this technique is quite straightforward and is thus a popular strategy for combining the predictive power of different recommendation techniques in a weighted manner.

Switching hybrids
Switching hybrids require an oracle that decides which recommender should be used in a specific situation, depending on the user profile and/or the quality of recommendation results. Such an evaluation could be carried out as follows:
∃1k : 1. . . n recswitching(u, i) = reck (u, i) where
k is determined by the switchingcondition.

Pipelined hybridization design

Pipelined hybrids implement a staged process in which several techniques sequentially build on each other before the final one produces recommendations for the user.

The pipelined hybrid variants differentiate themselves mainly according to the type of output they produce for the next stage. In other words, a preceding component may either preprocess input data to build a model that is exploited by the subsequent stage or deliver a recommendation list for furtherrefinement.

Cascade hybrids
Cascade hybrids are based on a sequenced order of techniques, in which each succeeding recommender only refines the recommendations of its predecessor.

The recommendation list of the successor technique is thus restricted to items that were also recommended by the preceding technique.

Formally, assume a sequence of n techniques, where rec1 represents the recommendation function of the first technique and recn the last one. Consequently, the final recommendation score for an item is computed by the nth technique. However, an item will be suggested by the kth technique only if the (k − 1)th technique also assigned a nonzero score to it. This applies to all k ≥ 2 by induction as defined inFormula

$$rec_{cascade}(u, i) = rec_n(u, i)$$

where $\forall k \geq 2$ must hold:

$$rec_k(u, i) = \begin{cases} rec_k(u, i) & : rec_{k-1}(u, i) \neq 0 \\ 0 & : \text{else} \end{cases}$$

Thus in a cascade hybrid all techniques, except the first one, can only change the ordering of the list of recommended items from their predecessor or exclude an item by setting its utility to 0

Meta-level hybrids

In a meta-level hybridization design, one recommender builds a model that is exploited by the principal recommender to make recommendations.

Formula that formalizes this behavior, wherein the nth recommender exploits a model that has been built by its predecessor. However, in all reported systems so far, n has always been 2.

$$rec_{meta-level}(u, i) = rec_n(u, i, \Delta_{rec_{n-1}})$$

UNIT 4

Stream Concepts

- Recently a new class of data-intensive applications has become widely recognized applications in which the data is modeled best not as persistent relations but rather as transient datastreams.
- Examples of such applications include financial applications, network monitoring, security, telecommunications data management, web applications, manufacturing, sensor networks, andothers.
- In the data stream model, individual data items may be relational tuples, e.g., network measurements, call records, web page visits, sensor readings, and so on.
  - However, their continuous arrival in multiple, rapid, time-varying, possibly unpredictable and unbounded streams appears to yield some fundamentally new researchproblems.

Data Stream Model

- A data stream is a real time continuous and ordered sequence of items. It is not possible to control the order in which the items arrive, nor it is feasible to locally store a stream in its entirety in any memorydevice.
- Further a query over streams will actually run continuously over a period of time and return new results as new dataarrives.
- Therefore these are known as long running, continuous, standing and persistentqueries.

Characteristics

1. The data model and query processor must allow both order based and time based operations.
2. The inability to store a complete stream indicates that some approximate summary structures must beused.
3. Streaming query plans must not use any operators that require the entire input before any results are produced. Such operators will block the query processor indefinitely.
4. Any query that requires backtracking over a data streams is infeasible. This is due to storage and performance constraints imposed by a datastream.
5. Applications that monitor streams in real time must react quickly to unusual data values.
6. Scalability requirements dictate the parallel and shared execution of many continuous queries must bepossible.

Architecture

An input monitor may regulate the input streams perhaps by dropping packets.
Data are typically stored in three partitions.

1. Temporary working storage (for windowqueries)
2. Summarystorage
3. Static storage for meta-data (Physical location of eachstorage)

- Long running queries are registered in the query repository and placed into groups for sharedprocessing.

- The query processor communicates with the input monitor and may reoptimize the query plans in response to changing input rates. Results are streamed to the user or temporarilybuffered.

- A Data-Stream-Management SystemIn analogy to a database-management system, we can view a stream processor as a kind of data-management system, the high-level organization.

Any number of streams can enter the system. Each stream can provide elements at its own schedule; they need not have the same data rates or data types, and the time between elements of one stream need not be uniform.

The fact that the rate of arrival of stream elements is not under the control of the system distinguishes stream processing from the processing of data that goes on within a database-managementsystem.

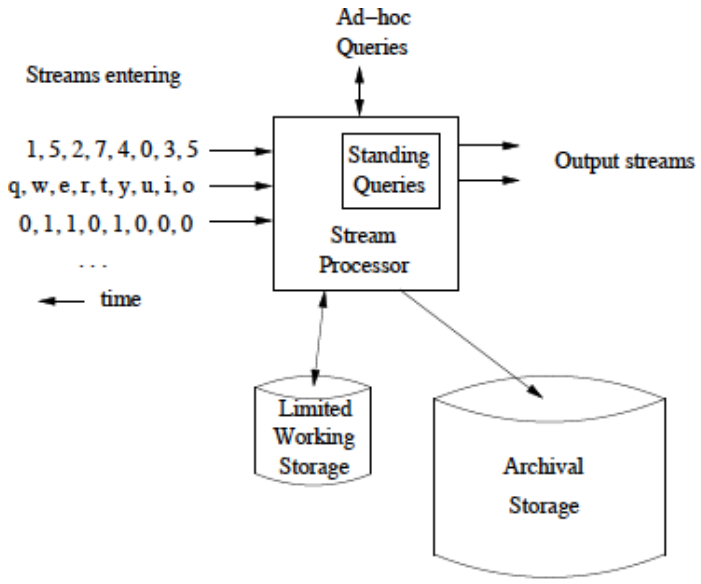


Figure 4.1: A data-stream-management system

- Streams may be archived in a large archival store, but we assume it is not possible to answer queries from the archival store. It could be examined only under special circumstances using time-consuming retrievalprocesses.
- There is also a working store, into which summaries or parts of streams may be placed, and which can be used for answeringqueries.
- The working store might be disk, or it might be main memory, depending on how fast we need to processqueries.

Examples of Data Stream Applications

Sensor Networks – SN are large source of data occurring in streams. Used in numerous situations that require constant monitoring of several variables based on which important decisions are made. Alerts and alarms generated as a response to information received from sensors. Example – Perform joins of several streams like temperature, ocean currents streams at weather stations to give alerts or warningslike

cyclone or tsunami.

Network Traffic Analysis – ISP's get information about Internet traffic, heavily used routes etc. to identify and predict congestions. Streams also used to identify fraudulent activities. Example queries – Check whether current stream of actions over time are similar to previous intrusion on thenetwork.

Financial Applications – Online analysis of stock prices and making hold or sell decisions requires quickly identifying correlations and fast changing trends.

Transaction Log Analysis – Online mining of web usage logs, telephone call records and ATM transactions are examples of data streams. Goal is to find customer behavior patterns. Example – Identify current buying pattern of users in website and plan advertising campaigns andrecommendations.

Image Data

Satellites often send down to earth streams consisting of many terabytes of images per day. Surveillance cameras produce images with lower resolution than satellites, but there can be many of them, each producing a stream of images at intervals like one second.

Stream Computing

Stream Queries

There are two ways that queries get asked about streams.

One-time queries - (a class that includes traditional DBMS queries) are queries that are evaluated once over a point-in-time snapshot of the data set, with the answer returned to the user. Example - Alert when stock crosses over a pricepoint.

Continuous queries, on the other hand, are evaluated continuously as data streams continue to arrive. Continuous query answers may be stored and updated as new data arrives or they may be produced as data streams themselves. Example –Aggregation queries like maximum, average, count etc. Maximum price of stock every hour, or number of time stock gains over a particular point.

Issues in Stream Processing

First, streams often deliver elements very rapidly. We must process elements in real time, or we lose the opportunity to process them at all, without accessing the archival storage.

Thus, it often is important that the stream-processing algorithm is executed in main memory, without access to secondary storage or with only rare accesses to secondary storage.

Thus, many problems about streaming data would be easy to solve if we had enough memory, but become rather hard and require the invention of new techniques in order to execute them at a realistic rate on a machine of realistic size.

Sampling Data in a Stream

As our first example of managing streaming data, we shall look at extracting reliable samples from a stream.

The general problem we shall address is selecting a subset of a stream so that we can ask queries about the selected subset and have the answers be statistically representative of the stream as a whole. If we know what queries are to be asked, then there are a number of methods that might work, but we are looking for a technique that will allow ad-hoc queries on thesample.

The General Sampling Problem

The running example is typical of the following general problem. Our stream consists of tuples with n components. A subset of the components are the key components, on which the selection of the sample will be based. In our running example, there are three components – user, query, and time – of which only user is in the key.

However, we could also take a sample of queries by making query be the key, or even take a sample of user-query pairs by making both those components form the key.

To take a sample of size a/b, we hash the key value for each tuple to b buckets, and accept the tuple for the sample if the hash value is less than a. If the key consists of more than one component, the hash function needs to combine the values for those components to make a single hash-value.

Varying the Sample Size

Often, the sample will grow as more of the stream enters the system. In our running example, we retain all the search queries of the selected 1/10th of the users, forever. As time goes on, more searches for the same users will be accumulated, and new users that are selected for the sample will appear in thestream.

If we have a budget for how many tuples from the stream can be stored as the sample, then the fraction of key values must vary, lowering as time goes on.

In order to assure that at all times, the sample consists of all tuples from a subset of the key values, we choose a hash function h from key values to a very large number of values 0, 1, . . . ,B−1. We maintain a threshold t, which initially can be the largest bucket number, B −1.

If the number of stored tuples of the sample exceeds the allotted space, we lower t to t−1 and remove from the sample all those tuples whose key K hashes to t.

For efficiency, we can lower t by more than 1, and remove the tuples with several of the highest hash values, whenever we need to throw some key values out of the sample.

Further efficiency is obtained by maintaining an index on the hash value, so we can find all those tuples whose keys hash to a particular value quickly.

Filtering Streams

One common process on streams is selection, or filtering. We want to accept those tuples in the stream that meet a criterion.

Accepted tuples are passed to another process as a stream, while other tuples are dropped. If the selection criterion is a property of the tuple that can be calculated (e.g., the first component is less than 10), then the selection is easy todo.

The problem becomes harder when the criterion involves lookup for membership in a set. It is especially hard, when that set is too large to store in main memory.

The Bloom Filter

A Bloom filter consists of:

1. An array of n bits, initially all0's.
2. A collection of hash functions h**1**, h**2**, . . . ,h**k**. Each hash function maps "key" values to n buckets, corresponding to the n bits of thebit-array.
3. A set S of m keyvalues.

The purpose of the Bloom filter is to allow through all stream elements whose keys are in S, while rejecting most of the stream elements whose keys are not in S.

To initialize the bit array, begin with all bits 0. Take each key value in S and hash it using each of the k hash functions. Set to 1 each bit that is h**i**(K) for some hash function h**i** and some key value K in S.

To test a key K that arrives in the stream, check that all of h**1**(K), h**2**(K), . . . ,h**k**(K)

are 1's in the bit-array. If all are 1's, then let the stream element through. If one or more of these bits are 0, then K could not be in S, so reject the stream element.

Analysis of Bloom Filtering

If a key value is in **S**, then the element will surely pass through the Bloom filter. However, if the key value is not in **S**, it might still pass. We need to understand how to calculate the probability of a false positive, as a function of **n**, the bit-array length, **m** the number of members of **S**, and **k**, the number of hash functions.

The model to use is throwing darts at targets. Suppose we have **x** targets and **y** darts. Any dart is equally likely to hit any target. After throwing the darts, how many targets can we expect to be hit at least once?

- The probability that a given dart will not hit a given target is (**x** −1)/**x**.
- The probability that none of the **y** darts will hit a given target is $\left(\frac{!!!}{}\right)^{y}$!
- Usingtheapproximation$(1-\in)^{1/e}$=1/eforsmall∈weconcludethattheprobability that noneof the **y** darts hit agiven targetis$^{!!/!}$.

Counting Distinct Elements in a Stream

The Count-Distinct Problem
Suppose stream elements are chosen from some universal set. We would like to know how many different elements have appeared in the stream, counting either from the beginning of the stream or from some known time in the past.

The Flajolet-Martin Algorithm

It is possible to estimate the number of distinct elements by hashing the elements of the universal set to a bit-string that is sufficiently long.

The length of the bit-string must be sufficient that there are more possible results of the hash function than there are elements of the universal set.

The idea behind the Flajolet-Martin Algorithm is that the more different elements we see in the stream, the more different hash-values.

As we see more different hash-values, it becomes more likely that one of these values will be "unusual."

The particular unusual property we shall exploit is that the value ends in many 0's, although many other options exist. Whenever we apply a hash function h to a stream element a, the bit string h(a) will end in some number of 0's, possibly none.

Call this number the tail length for a and h. Let R be the maximum tail length of any a seen so far in the stream. Then we shall use estimate 2^R for the number of distinct elements seen in the stream.

We can conclude:

1. If m is much larger than 2^r, then the probability that we shall find a tail of length at least r approaches1.
2. If m is much less than 2^r, then the probability of finding a tail length at least r approaches0.

We conclude from these two points that the proposed estimate of m, which is 2^R (recall R is the largest tail length for any stream element) is unlikely to be either much too high or much too low.

CombiningEstimates

Our first assumption would be that if we take the average of the values 2^R that we get from each hash function, we shall get a value that approaches the true m, the more hash functions weuse.

However, that is not the case, and the reason has to do with the influence an overestimate has on the average.

Another way to combine estimates is to take the median of all estimates. The median is not affected by the occasional outsized value of 2^R, so the worry described above for the average should not carry over to themedian.

Unfortunately, the median suffers from another defect: it is always a power of 2. Thus, no matter how many hash functions we use, should the correct value of m be between two powers of 2, say 400, then it will be impossible to obtain a close estimate.

There is a solution to the problem, however. We can combine the two methods. First, group the hash functions into small groups, and take their average. Then, take the median of the averages. It is true that an occasional outsized 2^R will bias some of the groups and make them too large.

However, taking the median of group averages will reduce the influence of this effect almost to nothing. In order to guarantee that any possible average can be obtained, groups should be of size at least a small multiple of log2 m.

Space Requirements

The stream it is not necessary to store the elements seen. The only thing we need to keep in main memory is one integer per hash function;

This integer records the largest tail length seen so far for that hash function and any stream element.

If we are processing only one stream, we could use millions of hash functions, which is far more than we need to get aclose estimate.

Only if we are trying to process many streams at the same time would main memory constrain the number of hash functions we could associate with any one stream.

In practice, the time it takes to compute hash values for each stream element would be the more significant limitation on the number of hash functions we use.

Estimating Moments

The problem, called computing "moments," involves the distribution of frequencies of different elements in thestream.

Suppose a stream consists of elements chosen from a universal set. Assume the universal set is ordered so we can speak of the ith element for any i. Let m_i be the number of occurrences of the ith element for any i. Then the kth-order moment (or just kth moment) of the stream is the sum over all i of$(mi)^{k.}$

The Alon-Matias-Szegedy Algorithm for Second Moments

For now, let us assume that a stream has a particular length n. Suppose we do not have enough space to count all the mi's for all the elements of thestream.

We can still estimate the second moment of the stream using a limited amount of space; the more space we use, the more accurate the estimate will be. We compute some number of variables. For each variable X, we store:

1. A particular element of the universal set, which we refer to as X.element ,and
2. An integer X.value, which is the value of the variable. To determine the value of a variable X, we choose a position in the stream between 1 and n, uniformly and at random. Set X.element to be the element found there, and initialize X.value to 1. As we read the stream, add 1 to X.value each time we encounter another occurrence of X.element.

Why the Alon-Matias-Szegedy Algorithm Works
We can prove that the expected value of any variable constructed is the second moment of the stream from which it is constructed.

Some notation will make the argument easier to follow. Let e(i) be the stream element that appears at position i in the stream, and let c(i) be the number of times element e(i) appears in the stream among positions i, i + 1, . . . , n.

Higher-Order Moments

We estimate kth moments, for k> 2, in essentially the same way as we estimate second moments.

The only thing that changes is the way we derive an estimate from a variable. We used the formula n(2v − 1) to turn a value v, the count of the number of occurrences of some particular stream element a, into an estimate of the secondmoment.

We saw why this formula works: the terms 2v − 1, for v = 1, 2, . . . ,m sum to m2, where m is the number of times a appears in the stream.

Counting Ones in a Window

Suppose we have a window of length N on a binary stream. We want at all times to be able to answer queries of the form "how many 1's are there in the last k bits?" for any k ≤ N .

As in previous sections, we focus on the situation where we cannot afford to store the entire window. After showing an approximate algorithm for the binary case, we discuss how this idea can be extended to summing numbers.

The Cost of Exact Counts

To begin, suppose we want to be able to count exactly the number of 1's in the last k bits for any k ≤ N. Then we claim it is necessary to store all N bits of the window, as any representation that used fewer than N bits could not work.

In proof, suppose we have a representation that uses fewer than N bits to represent the N bits in the window. Since there are 2^N sequences of N bits, but fewer than 2^N representations,

There must be two different bit strings w and x that have the same representation. Since w≠x, they must differ in at least one bit. Let the last k −1 bits of w and x agree, but let them differ on the kth bit from the right end.

The Datar-Gionis-Indyk-Motwani Algorithm

We shall present the simplest case of an algorithm called DGIM. This version of the algorithm uses $O(\log^2 N)$ bits to represent a window of N bits, and allows us to estimate the number of 1's in the window with an error of no more than 50%.

Later, we shall discuss an improvement of the method that limits the error to any fraction ǫ > 0, and still uses only $O(\log^2 N)$ bits (although with a constant factor that grows as ǫ shrinks).

To begin, each bit of the stream has a timestamp, the position in which it arrives. The first bit has timestamp 1, the second has timestamp 2, and so on.

Since we only need to distinguish positions within the window of length N, we shall represent timestamps modulo N, so they can be represented by $\log_2 N$ bits.

We divide the window into buckets,5 consisting of:

1. The timestamp of its right (most recent)end.
2. The number of 1's in the bucket. This number must be a power of 2, and we refer to the number of 1's as the size of thebucket.

There are six rules that must be followed when representing a stream by buckets.

- The right end of a bucket is always a position with a1.
- Every position with a 1 is in somebucket.
- No position is in more than onebucket.
- There are one or two buckets of any given size, up to some maximumsize.
- All sizes must be a power of2.
- Buckets cannot decrease in size as we move to the left (back intime).

Storage Requirements for the DGIMAlgorithm

We observed that each bucket can be represented by O(logN) bits. If the window has length N, then there are no more than N 1's,surely.

Suppose the largest bucket is of size 2j. Then j cannot exceed log2 N, or else there are more1's in this bucket than there are 1's in the entire window. Thus, there are at most two buckets of all sizes from log2 N down to 1, and no buckets of larger sizes.

We conclude that there are O(logN) buckets. Since each bucket can be represented in O(logN) bits, the total space required for all the buckets representing a window of size N is O(log2 N).

Query Answering in the DGIM Algorithm

Suppose we are asked how many 1's there are in the last k bits of the window, for some $1 \leq k \leq N$.

Find the bucket b with the earliest timestamp that includes at least some of the k most recent bits. Estimate the number of 1's to be the sum of the sizes of all the buckets to the right (more recent) than bucket b, plus half the size of b itself.

Decaying Windows

We have assumed that a sliding window held a certain tail of the stream, either the most recent N elements for fixed N, or all the elements that arrived after some time in the past.

Sometimes we do not want to make a sharp distinction between recent elements and those in the distant past, but want to weight the recent elements more heavily. In this section, we consider "exponentially decaying windows," and an application where they are quite useful: finding the most common "recent" elements.

The Problem of Most-Common Elements

Suppose we have a stream whose elements are the movie tickets purchased all over the world, with the name of the movie as part of the element. We want to keep a summary of the stream that is the most popular movies"currently."

On the other hand, a movie that sold n tickets in each of the last 10 weeks is probably more popular than a movie that sold 2n tickets last week but nothing in previous weeks.

One solution would be to imagine a bit stream for each movie. The ith bit has value 1 if the ith ticket is for that movie, and 0 otherwise. Pick a window size N, which is the number of most recent tickets that would be considered in evaluating popularity.

Definition of the Decaying Window

An alternative approach is to redefine the question so that we are not asking for a count of 1's in a window. Rather, let us compute a smooth aggregation of all the 1's ever seen in the stream, with decaying weights, so the further back in the stream, the less weight is given.

Formally, let a stream currently consist of the elements a1, a2, . . . , at, where a1 is the first element to arrive and at is the current element. Let c be a small constant, such as 10^{-6} or 10^{-9}. Define the exponentially decaying window for this stream to be the sum.

$$\sum_{i=0}^{t-1} a_{t-i}(1-c)^i$$

Finding the Most Popular Elements

Let us return to the problem of finding the most popular movies in a stream of ticket sales. We shall use an exponentially decaying window with a constant c, which you might think of as 10^{-9}. That is, we approximate a sliding window holding the last one billion ticket sales.

We imagine that the number of possible movies in the stream is huge, so we do not want to record values for the unpopular movies. Therefore, we establish a threshold, say 1/2, so that if the popularity score for a movie goes below this number, its score is dropped from the counting.

For reasons that will become obvious, the threshold must be less than 1, although it can be any number less than 1. When a new ticket arrives on the stream, do the following:

1. Foreachmoviewhosescorewearecurrentlymaintaining,multiplyitsscoreby(1 −c).
2. Suppose the new ticket is for movie M. If there is currently a score for M, add 1 to that score. If there is no score for M, create one and initialize it to1.
3. If any score is below the threshold 1/2, drop thatscore.

It may not be obvious that the number of movies whose scores are maintained at any time is limited. However, note that the sum of all scores is 1/c.

There cannot be more than 2/c movies with score of 1/2 or more, or else the sum of the scores would exceed1/c.

Thus, 2/c is a limit on the number of movies being counted at any time.

Real Time Analytical Platform

Real time analytics makes use of all available data and resources when they are needed. It consist of dynamic analysis and reporting based on the entered on to a system less than one minute before the actual time of use.

Real time denotes the ability to process as it arrives, rather than storing the data and retrieving it at some point in the future.

Real time analytics is thus delivering meaningful patterns in the data for something urgent. Types of real time analytics

On Demand Real Time Analytics – It is reactive because it waits for users to request a query and then delivers the analytics. This is used when someone within a company needs to take a pulse on what is happening right this minute.

Continuous Real Time Analytics – It is more proactive and alerts users with continuous updates in real time. Example Monitoring stock market trends provide analytics to help users make a decision to buy or sell all in real time.

Real Time Analytics Applications

Financial Services – Analyze tickets, tweets, satellite integrity, weather trends, and any other type of data to inform trading algorithm in realtime.

Government – Identify social program fraud within seconds based on program history, citizen profile, and geographicaldata.

E-Commerce sites – Real time analytics will help to tap into user preferences as people are on the site or using product. By knowing what user likes at a run time can help the site to decide relevant content to be made available to that user. This can result in better customer experience overall leading to increase in sales.

Insurance Industry – Digital channel of customers interaction as well as conversations online have created new stream of real time event data.

Generic Design of an RTAP

Companies like Facebook and twitter generates petabytes of real time data. This data must be harnessed to provide real time analytics to make better business decisions. Today Billions of devices are already connected to the internet with more connecting everyday.

Real time analytics will leverage information from all these devices to apply analytics algorithms and generate automated actions within milliseconds of a trigger.

Real time analytics needed the following aspects of data flow,

Input - An event happens (New sale, new customer, someone enters a high security zone etc.)

Process and Store Input – Capture the data of the event, and analyze the data without leveraging resources that are dedicated to operations.

Output – Consume this data without distributing operations

The following key capabilities must be provided by any analytical platform
- Delivering in Memory TransactionSpeed
- Quickly Moving Unneeded data to disk for long termstorage
- Distributing data and processing forspeed
- Supporting continuous queries for real timeevents
- Embedding data into apps or apps intodatabase
- Additionalrequirements

Many technologies support real time analytics, they are,
- Processing inmemory
- In databaseanalytics
- Data warehouseapplications
- In memoryanalytics
- Massive parallelprogramming

REAL TIME SENTIMENT ANALYSIS

Sentiment analysis also known as opinion mining refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials.

Sentiment analysis is widely applied to reviews and social media for a variety of applications ranging from marketing to customer service.

A basic task in sentiment analysis is classifying the polarity of a given text at the document, sentence, or feature/aspect level where the expressed opinion in a document, a sentence or an entity feature aspect is positive negative or neural.

Applications

News media website interested in getting edge over its competitors by featuring site content that is immediately relevant to its readers. They use social media analysis topics relevant to their readers by doing real time sentiment analysis on twitter data. Specifically to identify what topics are trending in real time on twitter.

Twitter has become a central site where people express their opinions and views on political parties and candidates. Emerging events or news or often followed almost instantly by a burst in twitter volume which if analyzed in real time can help explore how these events affect public opinion While traditional content analytics takes days or weeks to complete, RSTA can look into entire content about election and delivering results instantly andcontinuously.

Ad agencies can track the crowd sentiment during commercial viewing on TV and decide which commercials are resulting in positive sentiment and which are not.

Analyzing sentiments of messages posted to social media or online forums can generate countless business values for the organizations, which aims to extract timely business intelligence about how their products or services are perceived by their customers. As a result proactive marketing or product design strategies can be

developed to efficiently increase the customer base.

Tools

Apache Strom is a distributed real time computation system for processing large volume of data. It is part of Hadoop. Strom is extremely fast with the ability to process over a million of records per second per node on a cluster of modestsize.

Apache Solr is another tool from Hadoop which provides a highly reliable scalable search engine facility at real time.

RADAR is a software solution for retailers built using a Natural language processing based sentiment analysis engine and utilizing Hadoop technologies including HDFS, YARN, Apache Storm, Apache Solr, Oozie and Zookeeper to help them maximize sales through databased continuous repricing.

Online retailers can track the following for number of products in their portfolio,

a. Social sentiment for eachproduct
b. Competitive pricing / promotions being offered in social media and in theweb.

REAL TIME STOCK PREDICTION

Traditional stock market prediction algorithm check historical stock price and try to predict the future using different models. But in real time scenario stock market trends continually change economic forces, new products, competition, world events, regulations and even tweets are all factors to affect stock prices. Thus real time analytics to predict stock prices is the need of the hour.

A general real time stock prediction and machine learning architecture comprises three basiccomponents,

a. Incoming real time trading data must be captured and stored becoming historical data.
b. The system must be able to learn from historical trends in the data and recognize patterns and probabilities to informdecisions.
c. The system needs to do a real time comparison of new incoming trading data with the learned patterns and probabilities based in historical data. Then it predicts an outcome and determines a action totake.

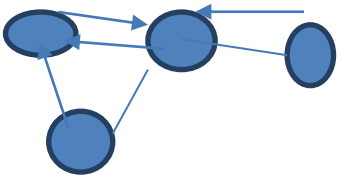
For Example consider the following,
Live data from Yahoo Finance or any other finance news RSS feed is real and processed. The data is then stored in memory with a fast consistent resilient and linearly scalable system.

Using the live hot data from Apache Geode, a Spark MLib application creates and trains a model computing new data to historical patterns. The models could also be supported by other toolsets such as Apache MADlib or R.

Results of the machine-learning model are pushed to other interested applications and also updated within Apache Geode for real time predication and decisioning.

As data ages and starts to become cool it is moved from Apache Geode to Apache HAWQ and eventually lands in Apache Hadoop. Apache HAWQ allows for SQL based analysis on petabyte scale data sets and allows data scientists to iterate on and improve models.

Another process is triggered to periodically retain and update the machine learning model based on the whole historical data set. This closes the loop and creates ongoing updates and improvements when historical patterns change or as new models emerge.



Using Graph Analytics for Big Data: Graph Analytics

WHAT IS GRAPH ANALYTICS?

Graph analytics is based on a model of representing individual entities and numerous kinds of relationships that connect those entities.

More precisely, it employs the graph abstraction for representing connectivity, consisting of a collection of vertices (which are also referred to as nodes or points) that represent the modeled entities, connected by edges (which are also referred to as links, connections, or relationships) that capture the way that two entities arerelated.

Among other enhancements, these can enrich the meaning of the nodes and edges represented in the graph model:

- Vertices can be labeled to indicate the types of entities that arerelated.
- Edges can be labeled with the nature of therelationship.
- Edges can be directed to indicate the "flow" of therelationship.
- Weights can be added to the relationships represented by theedges.
- Additional properties can be attributed to both edges andvertices.
- Multiple edges can reflect multiple relationships between pairs ofvertices

Representation as triples

In essence, these enhancements help in building a semantic graph—a directed graph that can be represented using a triple format consisting of a **subject** (the source point of the relationship), **an object** (the target), and **a predicate** (that models the type of therelationship).

A collection of these triples is called a semantic database, and this kind of database can capture additional properties of each triple relationship as attributes of the triple.

Graphs and network organization

One of the benefits of the graph model is the ability to detect patterns or organization that are inherent within the represented network, such as:

Embedded micronetworks: Looking for small collections of entities that form embedded "micro-communities."

Communication models: Modeling communication across a community triggered by a specific event, such as monitoring the "buzz" across a social media channel associated with the rumored release of a new product, evaluating best methods for communicating news releases, or correlation between travel delays and increased mobile telephony activity.

Collaborative communities: Isolating groups of individuals that share similar interests, such as groups of health care professionals working in the same area of specialty, purchasers with similar product tastes, or individuals with a precise set of employment skills.
Influence modeling: Looking for entities holding influential positions within a network for intermittent periods of time, such as computer nodes that have been hijacked and put to work as proxies for distributed denial of service attacks or for emerging cybersecurity threats, or individuals that are recognized as authorities within a particular area.
Distance modeling: Analyzing the distances between sets of entities, such as looking for strong correlations between occurrences of sets of statistically improbable phrases among large sets of search engines queries, or the amount of effort necessary to propagate a message among a set of different communities.

Choosing graph analytics

Deciding the appropriateness of an analytics application to a graph analytics solution instead of the other big data alternatives can be based on these characteristics and factors of business problems:

Connectivity, Undirected discovery, Absence of structure, Flexible semantics Extensibility, Knowledge is embedded in the network, Ad hoc nature of the analysis, Predictable interactive performance.

GRAPH ANALYTICS ALGORITHMS AND SOLUTION APPROACHES

The graph model is inherently suited to enable a broad range of analyses that are generally unavailable to users of a standard data warehouse framework.

Some of the types of analytics algorithmic approaches include:
Community and network analysis, in which the graph structures are traversed in search of groups of entities connected in particularly "close" ways. One example is a collection of entities that are completely connected (i.e., each member of the set is connected to all other members of the set).

Path analysis, which analyze the shapes and distances of the different paths that connect entities within the graph.

Clustering, which examines the properties of the vertices and edges to identify characteristics of entities that can be used to group them together.

Pattern detection and pattern analysis, or methods for identifying anomalous or unexpected patterns requiring further investigation.

Probabilistic graphical models such as Bayesian networks or Markov networks for various application such as medical diagnosis, protein structure prediction, speech recognition, or assessment of default risk for credit applications.

Graph metrics that are applied to measurements associated with the network itself, including the degree of the vertices (i.e., the number of edges in and out of the vertex), or centrality anddistance.

Technical complexity of analyzing graphs

There are some characteristics of graphs that inhibit the ability of typical computing platforms to provide the rapid responses or satisfy the need for scalability, especially as data volumes continue to explode. Some of the factors that might introduce performance penalties that are not easily addressed on standard hardware architecturesinclude:

Unpredictability of graph memory accesses: The types of discovery analyses in graph analytics often require the simultaneous traversal of multiple paths within a network to find the interesting patterns for further review or optimal solutions to aproblem.

Graph growth models: As more information is introduced into an environment, real-world networks grow in an interesting way.

Dynamic interactions with graphs: As with any big data application, graphs to be analyzed are populated from a wide variety of data sources of large or massive data volumes, streamed at varying rates.

Complexity of graph partitioning: Graphs tend to cluster among centers of high connectivity, as many networks naturally have "hubs" consisting of a small group of nodes with many connections.

Features of a graph analytics platform

Seamless data intake: Providing a seamless capability to easily absorb and fuse data from a variety of different sources.

Data integration: A semantics-based approach is necessary for graph analytics to integrate different sets of data that do not have predetermined structure.

Inferencing: The application platform should provide methods for inferencing and deduction of new information and insights derived from the embedded relationships and connectivity.

Standards-based representation: Any graph analytics platform must employ the resource description framework standard to use triples for representing the graph.

Workflow integration: Providing a graph analytics platform that is segregated from the existing reporting and analytics environments will have limited value when there are gaps in incorporating results from across the differentenvironments.

Visualization: Presenting discoveries using visualization tools is critical to highlight their value. Look for platforms that have tight integration with visualization services.

"Complementariness": A graph analytics platform augments an organization's analytics capability and is not intended to be a replacement.

UNIT 5

NoSQL Databases

WHAT IS NOSQL?

NoSQL (Not only Structured Query Language) is a term used to describe those data stores that are applied to unstructured data.

The term "NoSQL" may convey two different connotations—one implying that the data management system is not an SQL-compliant one, while other is "Not onlySQL,"suggestingenvironmentsthatcombinetraditionalSQL(orSQL-like querylanguages)withalternativemeansofqueryingandaccess.

Schema-less Models: Increasing Flexibility for Data Manipulation-Key ValueStores

NoSQL data systems hold out the promise of greater flexibility in database management while reducing the dependence on more formal database administration.

NoSQL databases have more relaxed modeling constraints, which may benefit both the application developer and the end-user.

Different NoSQL frameworks are optimized for different types of analyses.

In fact, the general concepts for NoSQL include schemaless modeling in which the semantics of the data are embedded within a flexible connectivity and storage model;

Thisprovidesforautomaticdistributionofdataandelasticitywithrespecttothe use of computing, storage, and network bandwidth in ways that don't force specificbindingofdatatobepersistentlystoredinparticularphysicallocations.

NoSQLdatabasesalsoprovideforintegrateddatacachingthathelpsreducedata access latency and speedperformance.

Thelooseningoftherelationalstructureisintendedtoallowdifferentmodelsto be adapted to specific types ofanalyses

Types of NoSql Key
Value Stores
Document Stores
Tabular Stores
Object Data Stores
Graph Databases

KEY VALUE STORES

Key/valuestorescontaindata(thevalue)thatcanbesimplyaccessedbyagiven identifier.
It is a schema-less model in which values (or sets of values, or even more complexentityobjects)areassociatedwithdistinctcharacterstringscalledkeys.

In a key/value store, there is no stored structure of how to use the data; the client that reads and writes to a key/value store needs to maintain and utilize thelogicofhowtomeaningfullyextracttheusefulelementsfromthekeyandthe value.

The key value store does not impose any constraints about data typing or data structure—the value associated with the key is the value.

The core operations performed on a key®value store include:

- Get(key),whichreturnsthevalueassociatedwiththeprovidedkey.
- Put(key,value),whichassociatesthevaluewiththekey.
- Multi-get(key1,key2,..,keyN),whichreturnsthelistofvaluesassociatedwith the list ofkeys.
- Delete(key),whichremovestheentryforthekeyfromthedatastore.

Keyvaluestoresareessentiallyverylong,andpresumablythintables.Thekeys can be hashed using a hash function that maps the key to a particular location (sometimescalleda“bucket”)inthetable.

The simplicity of the representation allows massive amounts of indexed data valuestobeappendedtothesamekey®valuetable,whichcanthenbesharded, or distributed across the storagenodes.

Drawbacks of Key Value Store
Oneisthatthemodelwillnotinherentlyprovideanykindoftraditionaldatabase capabilities (such as atomicity of transactions, or consistency when multiple transactionsareexecutedsimultaneously)—thosecapabilitiesmustbeprovided by the applicationitself.

Another is that as the model grows, maintaining unique values as keys may become more difficult, requiring the introduction of some complexity in generating character strings that will remain unique among a myriad of key.

DOCUMENT STORES

A document store is similar to a key value store in that stored objects are associated (and therefore accessed via) character string keys. The difference is thatthevaluesbeingstored,whicharereferredtoas“documents,”providesome structureandencodingofthemanageddata.

There are different common encodings, including XML (Extensible Markup Language),JSON(JavaScriptObjectNotation),BSON(whichisabinaryencoding ofJSONobjects),orothermeansofserializingdata.

Documentstoresareusefulwhenthevalueofthekey/valuepairisafileandthe file itself isself-describing.

One of the differences between a key®value store and a document store is that while the former requires the use of a key to retrieve data, the latter often providesameans(eitherthroughaprogrammingAPIorusingaquerylanguage) forqueryingthedatabasedonthecontents.

TABULAR STORES

Tabular, or table-based stores are largely descended from Google's original Bigtable design to manage structured data.

TheHBasemodelisanexampleofaHadoop-relatedNoSQLdatamanagement system that evolved frombigtable.

The bigtable NoSQL model allows sparse data to be stored in a three-dimensional table that is indexed by a row key, a column key that indicates the specificattributeforwhichadatavalueisstored,andatimestampthatmayrefer tothetimeatwhichtherow'scolumnvaluewasstored.

OBJECT DATA STORES

Insomeways,objectdatastoresandobjectdatabasesseemtobridgetheworlds ofschema-lessdatamanagementandthetraditionalrelationalmodels.

On the one hand, approaches to object databases can be similar to document storesexceptthatthedocumentstoresexplicitlyserializestheobjectsothedata values are stored as strings, while object databases maintain the object structuresastheyareboundtoobject-orientedprogramminglanguagessuchas C++, Objective-C, Java, andSmalltalk.

On the other hand, object database management systems are more likely to provide traditional ACID (atomicity, consistency, isolation, and durability) compliance—characteristics that are bound to database reliability.

Object databases are not relational databases and are not queried using SQL

GRAPH DATABASES

Graph databases provide a model of representing individual entities and numerous kinds of relationships that connect those entities.

More precisely, it employs the graph abstraction for representing connectivity, consisting of a collection of vertices (which are also referred to as nodes or points)thatrepresentthemodeledentities,connectedbyedges(whicharealso referredtoaslinks,connections,orrelationships)thatcapturethewaythattwo entities arerelated.

Graph analytics performed on graph data stores are somewhat different than more frequently used querying and reporting.

HIVE

Hive is a data warehouse infrastructure tool to process structured data in Hadoop.ItresidesontopofHadooptosummarizeBigData,andmakesquerying and analyzingeasy.

Hive facilitates easy data summarization, ad-hoc queries, and the analysis of large datasets stored in Hadoop compatible file systems."

Hiveisspecificallyengineeredfordatawarehousequeryingandreportingandis notintendedforuseaswithintransactionprocessingsystemsthatrequirereal-timequeryexecutionortransactionsemanticsforconsistencyattherowlevel.

Hive runs SQL like queries called HQL (Hive query language) which gets internally converted to map reduce jobs.

TheHivesystemprovidestoolsforextracting/transforming/loadingdata(ETL) intoavarietyofdifferentdataformats.

InitiallyHivewasdevelopedbyFacebook,latertheApacheSoftwareFoundation took it up and developed it further as an open source under the name Apache Hive.

Itisusedbydifferentcompanies.Forexample,AmazonusesitinAmazonElastic MapReduce.

Features of Hive

- ItstoresschemainadatabaseandprocesseddataintoHDFS.
- It is designed forOLAP.
- ItprovidesSQLtypelanguageforqueryingcalledHiveQLorHQL.
- Itisfamiliar,fast,scalable,andextensible.

Architecture of Hive

The following component diagram depicts the architecture of Hive:

User Interface

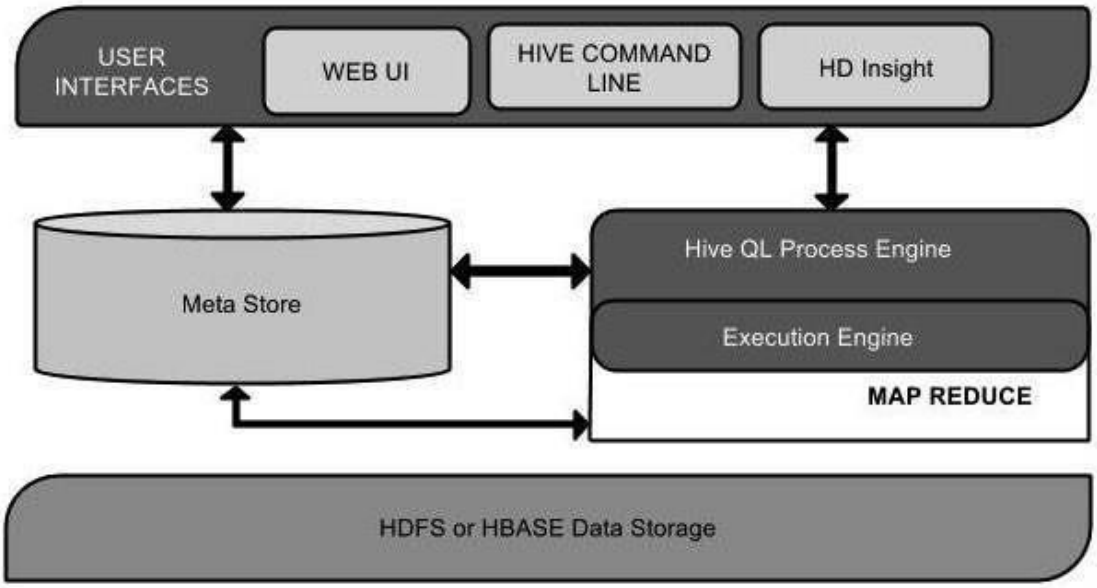
Hive is a data warehouse infrastructure software that can create interaction betweenuserandHDFS.TheuserinterfacesthatHivesupportsareHiveWebUI, Hivecommandline,andHiveHDInsight(InWindowsserver).

Meta Store

Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.

HiveQL Process Engine

HiveQLissimilartoSQLforqueryingonschemainfontheMetastore.Itisone ofthereplacementsoftraditionalapproachforMapReduceprogram.Insteadof writing MapReduce program in Java, we can write a query for MapReduce job and processit.



Execution Engine

The conjunction part of HiveQL process Engine and MapReduce is Hive ExecutionEngine.Executionengineprocessesthequeryandgeneratesresultsas sameasMapReduceresults.ItusestheflavorofMapReduce.

HDFS or HBASE

Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

Sharding

Shardingisadatabasearchitecturepatternrelatedtohorizontalpartitioning— thepracticeofseparatingonetable'srowsintomultipledifferenttables,known aspartitions.Eachpartitionhasthesameschemaandcolumns,butalsoentirely differentrows.

Databaseshardingisatypeofhorizontalpartitioningthatsplitslargedatabases intosmallercomponents,whicharefasterandeasiertomanage.

A shard is an individual partition that exists on separate database server instance to spread load.

Auto sharding or data sharding is needed when a dataset is too big to bestored in a singledatabase.

As both the database size and number of transactions increase, so does the response time for querying the database. Costs associated with maintaining a huge database can also skyrocket due to the number and quality of computers you need to manage your workload.

Datashards,ontheotherhand,havefewerhardwareandsoftwarerequirements andcanbemanagedonlessexpensiveservers.

In a vertically-partitioned table, entire columns are separated out and put into new, distinct tables. The data held within one vertical partition is independent from the data in all the others, and each holds both distinct rows and columns.

Shardinginvolvesbreakingupone'sdataintotwoormoresmallerchunks,called logicalshards.

Thelogicalshardsarethendistributedacrossseparatedatabasenodes,referred toasphysicalshards,whichcanholdmultiplelogicalshards.

ShardingArchitecture s Key Based Sharding

Keybasedsharding,alsoknownashashbasedsharding,involvesusingavalue taken from newly written data — such as a customer's ID number, a client application'sIPaddress,aZIPcode,etc.—andpluggingitintoahashfunctionto determinewhichshardthedatashouldgoto.

A hash function is a function that takes as input a piece of data (for example, a customer email) and outputs a discrete value, known as a hash value.

To ensure that entries are placed in the correct shards and in a consistent manner,thevaluesenteredintothehashfunctionshouldallcomefromthesame column.Thiscolumnisknownasas*hardkey*.

RANGE BASED SHARDING

Range based sharding involves sharding data based on ranges of a given value.

The main benefit of range based sharding is that it's relatively simple to implement.Everyshardholdsadifferentsetofdatabuttheyallhaveanidentical schemaasoneanother,aswellastheoriginaldatabase.

On the other hand, range based sharding doesn't protect data from being unevenly distributed, leading to the aforementioned database hotspots.

Directory Based Sharding

Toimplementdirectorybasedsharding,onemustcreateandmaintainalookup tablethatusesashardkeytokeeptrackofwhichshardholdswhichdata.

The main appeal of directory based sharding is its flexibility. Range based sharding architectures limit you to specifying ranges of values, while keybased oneslimityoutousingafixedhashfunctionwhich,asmentionedpreviously,can beexceedinglydifficulttochangelateron.

Directorybasedsharding,ontheotherhand,allowsyoutousewhateversystem oralgorithmyouwanttoassigndataentriestoshards,andit'srelativelyeasyto dynamically add shards using thisapproach.

While directory based sharing is the most flexible of the sharing methods discussed here, the need to connect to the lookup table before every query or write can have a detrimental impact on an application's performance.

HBASE

HBaseisano relationaldatamanagementenvironmentthatdistributesmassive datasets over the underlying Hadoopframework.

HBase is derived from Google's BigTable and is a column-oriented data layout that, when layered on top of Hadoop, provides a fault-tolerant method for storing and manipulating large data tables.

Data stored in a columnar layout is amenable to compression, which increases the amount of data that can be represented while decreasing the actual storage footprint.

In addition, HBase supports in-memory execution. HBase is not a relational database, and it does not support SQL queries.

There are some basic operations for Base:Get (which access a specific row in the table), Put(whichstoresorupdatesarowinthetable),Scan(whichiteratesoveracollectionofrowsinthetable),and Delete(whichremovesarowfromthetable).

Because it can be used to organize datasets, coupled with the performance provided by the aspects of the columnar orientation, Base is a reasonable alternative as a persistent storage paradigm when running Map Reduceapplications.

Review of Basic Data Analytic Methods using R.

Risaprogramminglanguageandsoftwareframeworkforstatisticalanalysisand graphics.

ThefollowingRcodeillustratesatypicalanalyticalsituationinwhichadatasetis imported,thecontentsofthedatasetareexamined,andsomemodelingbuilding tasks areexecuted.

```
#importaCSVfileofthetotalannualsalesforeachcustomer sales <-
read.csv("c:/data/yearly_sales.csv")
#examinetheimporteddataset head(sales)
summary(sales)
# plot num_of_orders vs. sales
plot(sales$num_of_orders,sales$sales_total, main="Number of Orders
vs. Sales")
#performastatisticalanalysis(fitalinearregressionmodel) results <-
lm(sales$sales_total~ sales$num_of_orders) summary(results)
#performsomediagnosticsonthefittedmodel # plot histogram of
the residuals hist(results$residuals, breaks =800)
```

Inthisexample,thedatafileisimportedusingtheread.csv()function.Oncethe file has been imported, it is useful to examine the contents to ensure that the data was loaded properly as well as to become familiar with the data. In the example,thehead()function,bydefault,displaysthefirstsixrecordsofsales.

The summary() function provides some descriptive statistics, such as the mean and median, for each data column.

Plotting a dataset's contents can provide information about the relationships between the various columns. In this example, the plot() function generates a scatterplot of the number of orders (sales$num_of_orders) against the annual sales (sales$sales_total)

Thesummary()functionisanexampleofagenericfunction.Agenericfunctionis agroupoffunctionssharingthesamenamebutbehavingdifferentlydepending onthenumberandthetypeofargumentstheyreceive.

Data Import and Export

In the annual retail sales example, the dataset was imported into R using the read.csv() function as in the following code.

```
sales <- read.csv("c:/data/yearly_sales.csv")
```

R uses a forward slash (/) as the separator character in the directory and file paths.

Otherimportfunctionsincluderead.table()andread.delim(),whichareintended toimportothercommonfiletypessuchasTXT.Thesefunctionscanalsobeused toimporttheyearly_sales.csvfile,asthefollowingcodeillustrates.

```
sales_table<-read.table("yearly_sales.csv",header=TRUE,sep=",") sales_delim<-
read.delim("yearly_sales.csv",sep=",")
```

The main difference between these import functions is the default values. For example, the read .delim() function expects the column separator to be a tab ("\t").

The analogous R functions such as write.table(), write.csv(), and write.csv2() enable exporting of R datasets to an external file. For example, the following R code adds an additional column to the sales dataset and exports the modified dataset to an external file.

```
# add a column for the average sales per order sales$per_order<-
sales$sales_total/sales$num_of_orders#exportdataastabdelimitedwithouttherow
names
write.table(sales,"sales_modified.txt", sep="\t", row.names=FALSE
```

**********ALL THE BEST**********