Module 1

Q.1.Give difference between Traditional data management and analytics approach Versus Big data Approach

Tradational Data	Big Data
Traditional data is generated in enterprise level.	Big data is generated outside the enterprise level.
Its volume ranges from Gigabytes to Terabytes.	Its volume ranges from Petabytes to Zettabytes or Exabytes.
Traditional database system deals with structured data.	Big data system deals with structured, semi-structured,database, and unstructured data.
Traditional data is generated per hour or per day or more.	But big data is generated more frequently mainly per seconds.
Traditional data source is centralized and it is managed in centralized form.	Big data source is distributed and it is managed in distributed form.
Data integration is very easy.	Data integration is very difficult.
Normal system configuration is capable to process traditional data.	High system configuration is required to process big data.
The size of the data is very small.	The size is more than the traditional data size.
Traditional data base tools are required to perform any data base operation.	Special kind of data base tools are required to perform any databaseschema-based operation.
Normal functions can manipulate data.	Special kind of functions can manipulate data.
Its data model is strict schema based and it is static.	Its data model is a flat schema based and it is dynamic.
Traditional data is stable and inter relationship.	Big data is not stable and unknown relationship.
Traditional data is in manageable volume.	Big data is in huge volume which becomes unmanageable.
It is easy to manage and manipulate the data.	It is difficult to manage and manipulate the data.
Its data sources includes ERP transaction data, CRM transaction data, financial data, organizational data, web transaction data etc.	Its data sources includes social media, device data, sensor data, video, images, audio etc.

Q2.What is Hadoop? Explain Hadoop ecosystem with core components. Explain physical architecture of hadoop. State its limitations

Hadoop ecosystem is a platform or framework which helps in solving the big data problems. It comprises of different components and services (ingesting, storing, analyzing, and maintaining) inside of it. The main four core components of Hadoop which include HDFS, YARN, MapReduce and Spark Zookeeper

Components of Hadoop:

- **HDFS** is the primary or major component of Hadoop ecosystem and is responsible for storing large data sets of structured or unstructured data
- **Yet Another Resource Negotiator(YARN)**, as the name implies, YARN is the one who helps to manage the resources across the clusters. In short, it performs scheduling and resource allocation for the Hadoop System.
- **MapReduce:** By making the use of distributed and parallel algorithms, MapReduce makes it possible to carry over the processing's logic and helps to write applications which transform big data sets into a manageable one.
    - Map() performs sorting and filtering of data and thereby organizing them in the form of group.
    - Reduce(), as the name suggests does the summarization by aggregating the mapped data.

Physical Architecture Of Hadoop:-

Name Node

- It is the master of HDFS (Hadoop file system).
- Contains Job Tracker, which keeps tracks of a file distributed to different data nodes.
- Failure of Name Node will lead to the failure of the full Hadoop system.

Data node

- Data node is the slave of HDFS.
- A data node can communicate with each other through the name node to avoid replication in the provided task.
- Data nodes update the change to the data node.

Job Tracker

- Determines which file to process.
- There can be only one job tracker for per Hadoop cluster.
- Only single task tracker is present per slave node.
- Performs tasks given by job tracker and also continuously communicates with the job tracker.

SSN (Secondary Name Node)

- Its main purpose is to monitor.
- One SSN is present per cluster

Limitations of Hadoop

- Issue with Small Files
- Slow Processing Speed

- Support for Batch Processing only
- No Real-time Data Processing

Q.3. How big data problems are handled by Hadoop system.

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models.

- It can handle arbitrary text and binary data.
- So Hadoop can digest any unstructured data easily.
- We saw in traditional approach having separate storage and processing clusters is not the best fit for big data. Hadoop clusters, however, provide storage and distributed computing all in one.

Q.4. Explain how Hadoop goals are covered in hadoop distributed file system.

Features of HDFS:

- **Highly Scalable** - HDFS is highly scalable as it can scale hundreds of nodes in a single cluster.
- **Replication** - Due to some unfavorable conditions, the node containing the data may be loss. So, to overcome such problems, HDFS always maintains the copy of data on a different machine.
- **Fault tolerance** - In HDFS, the fault tolerance signifies the robustness of the system in the event of failure. The HDFS is highly fault-tolerant that if any machine fails, the other machine containing the copy of that data automatically become active.
- **Distributed data storage** - This is one of the most important features of HDFS that makes Hadoop very powerful. Here, data is divided into multiple blocks and stored into nodes.
- **Portable** - HDFS is designed in such a way that it can easily portable from platform to another.

Goals of HDFS:

- **Handling the hardware failure** - The HDFS contains multiple server machines. Anyhow, if any machine fails, the HDFS goal is to recover it quickly.
- **Streaming data access -** The HDFS applications usually run on the general-purpose file system. This application requires streaming access to their data sets.
- **Coherence Model** - The application that runs on HDFS require to follow the write-once-read-many approach. So, a file once created need not to be changed. However, it can be appended and truncate.

Q 5. How Big data analytics can be useful in the development of Digital India?
Business Analytics, which is a mix of current tools, analytics, programming, business administration, and IT is very much in demand. Business Analytics helps us to improve upon existing data, secure them and make them available for future use in a better and more productive way.

Finance, media communications, outsourcing companies, internet business companies are some Indian sectors that make use of Business Analytics. Banks use data mining methods to filter the populated data and break down the accessible data using a few devices to identify the potential hazard sections and helps them to minimize the risks. MasterCard companies use Business Analytics to keep fake account activities at bay.

Data security is a major concern for many companies. Many issues with data security can be solved with Data Analytics. In India, resources may be utilized to garner data from deals agents, producers from the gear business, insurance agencies, by investigating the client data and securing it.

Data is distributed and scattered and needs to be segregated systematically through Business Analytics and standardize organizational structure. Many new businesses in India are using Big Data to pull out the essential and useful data. This has improved overall productivity and growth for the companies as well as their employees.

Big Data and Analytics solutions help businesses formulate better strategies devised on the back of data and logic. By providing insights on customers and prospective clients, their behavior, it allows companies to make meaningful, strategic adjustments that minimize costs and maximize results.

Q 6. Describe any 5 characteristics of Big Data.

1. **Volume**:The name 'Big Data' itself is related to a size which is enormous. Volume is a huge amount of data. To determine the value of data, size of data plays a very crucial role. If the volume of data is very large then it is actually considered as a 'Big Data'.
2. **Velocity**:Velocity refers to the high speed of accumulation of data. In Big Data velocity data flows in from sources like machines, networks, social media, mobile phones etc.
3. **Variety:** It refers to nature of data that is structured, semi-structured and unstructured data. It also refers to heterogeneous sources.
4. **Veracity**: It refers to inconsistencies and uncertainty in data, that is data which is available can sometimes get messy and quality and accuracy are difficult to control.
5. **Value**: After having the 4 V's into account there comes one more V which stands for Value!. The bulk of Data having no Value is of no good to the company, unless you turn it into something useful.

Q 7 What are the technical Challenges of Big Data?

Challenge #1 : Storing huge quantities of data.

- No storage machine is big enough to store the relentlessly growing quantity of data. Need to store in a large number of smaller inexpensive machines.
- There is the inevitable challenge of machine failure. Failure of a machine could entail a loss of data stored on it.

Solution: Distribute data across a large scalable cluster of inexpensive commodity machines,

- Ensures that every piece of data is systematically replicated on multiple machines to guarantee that at least one copy is always available,
- Add more machines as needed
- Hadoop Distributed File System is popular for managing large volumes

Challenge #2: Ingesting and processing streams at a fast pace

- Unpredictable and torrential streams of data too large to store, but must still be monitored.

Solution: Creating fast scalable ingesting and processing systems

- Can open an unlimited number of channels for receiving data Data can be held in queues, from Which business applications can read and process data at their own pace and convenience Apache Kafka is a popular dedicated ingest system
- The stream processing engine can do its work while the batch processing does its work. Apache Spark is the most popular system for streaming applications.

Challenge #3: Handling a variety of forms and functions of data

- Storing them in traditional flat or relational structures would be too impractical, wasteful and slow.
- Accessing and analyzing them requires different capabilities.

Solution: Use non relational systems that relax many of the stringent conditions of the relational model.

- These are called NoSQL (Not only SOL) databases, These databases optimized for certain tasks such as query processing, or graph processing document processing, etc
- HBase and Cassandra are two of the better known NoSQL databases

Challenge #4: Processing the huge quantities of data

- Moving large amounts of data from storage to the processor would consume enormous network capacity and choke the network.

Solution: Move the processing to where the data is stored.

- Distribute the task logic throughout the cluster of machines where the data is stored, Machines work, in parallel, on the data assigned to them
- A follow-up process consolidates intermediate outputs and delivers final results.

Module 2

Q1 What is the role of Job Tracker and Task Tracker in Map Reduce?

Job Tracker :- Job tracker is a daemon that runs on a namenode for submitting and tracking MapReduce jobs in Hadoop. It assigns the tasks to the different task tracker. In a Hadoop cluster, there will be only one job tracker but many task trackers. It is the single point of failure for Hadoop and MapReduce Service. If the job tracker goes down all the running jobs are halted. It receives heartbeat from task tracker based on which Job tracker decides whether the assigned task is completed or not.

Task Tracker :- Task tracker is also a daemon that runs on datanodes. Task Trackers manage the execution of individual tasks on slave node. When a client submits a job, the job tracker will initialize the job and divide the work and assign them to different task trackers to perform MapReduce tasks. While performing this action, the task tracker will be simultaneously communicating with job tracker by sending heartbeat. If the job tracker does not receive heartbeat from task tracker within

Q2 Give Map Reduce algorithm to perform relational algebra operations, selection, projection, unionand intersection of two sets.

Relational-Algebra Operations

- R, S - relation
- t, t' - a tuple
- C - a condition of selection
- A, B, C - subset of attributes
- a, b, c - attribute values for a given subset of attributes

Selection

- Map: For each tuple t in R, test if it satisfies C. If so, produce the key-value pair (t, t). That is, both the key and value are t.
- Reduce: The Reduce function is the identity. It simply passes each key-value pair to the output.

Projection

- Map: For each tuple t in R, construct a tuple t' by eliminating from t those components whose attributes are not in A. Output the key-value pair (t', t').
- Reduce: For each key t' produced by any of the Map tasks, there will be one or more key-value pairs (t',t').

Union

- Map: Turn each input tuple t either from relation R or S into a key-value pair (t, t).
- Reduce: Associated with each key t there will be either one or two values. Produce output (t, t) in either case.

Intersection

- Map: Turn each input tuple t either from relation R or S into a key-value pair (t, t).
- Reduce: If key t has value list [t, t], then produce (t,t). Otherwise, produce nothing.

Q3 Explain the concept of Map Reduce

MapReduce is a Hadoop framework and programming model for processing big data using automatic parallelization and distribution in the Hadoop ecosystem.

MapReduce consists of two essential tasks, i.e., Map and Reduce.

The reduce task always follows the map task. In the Map task, data are divided into chunks and processed in parallel.

The output of a map task is used as input in reduce tasks, and the data is shuffled and reduced.

- **Map()** performs sorting and filtering of data and thereby organizing them in the form of group.
- **Reduce(),** as the name suggests does the summarization by aggregating the mapped data.

MapReduce makes it possible to carry over the processing's logic and helps to write applications which transform big data sets into a manageable one.

MapReduce process has the following phases:

- **Input Splits**:- MapReduce splits the input into smaller chunks called input splits.
- **Mapping**:- The input data is processed and divided into smaller segments in the mapper phase, where the number of mappers is equal to the number of input splits.
- **Shuffling**:- In the shuffling phase, the output of the mapper phase is passed to the reducer phase by removing duplicate values and grouping the values.
- **Sorting**:- The Sorting phase involves merging and sorting the output generated by the mapper.
- **Reducing**:- In the reducer phase, the intermediate values from the shuffling phase are reduced to produce a single output value that summarizes the entire dataset.

Module 3

Q1. Why NoSQL is better than SQL OR Explain different ways by which big data problems are handled by NoSQL OR Differentiate between RDBMS and NoSQL database

NoSQL databases offer many benefits over relational databases. NoSQL databases have flexible data models, scale horizontally, have incredibly fast queries, and are easy for developers to work with.

Flexible data models: NoSQL databases typically have very flexible schemas. A flexible schema allows you to easily make changes to your database as requirements change. You can iterate quickly and continuously integrate new application features to provide value to your users faster.

Horizontal scaling: Most SQL databases require you to scale-up vertically (migrate to a larger, more expensive server) when you exceed the capacity requirements of your current server. Conversely, most NoSQL databases allow you to scale-out horizontally, meaning you can add cheaper commodity servers whenever you need to.

Fast queries: Queries in NoSQL databases can be faster than SQL databases. Why? Data in SQL databases is typically normalized, so queries for a single object or entity require you to join data from multiple tables. As your tables grow in size, the joins can become expensive. However, data in NoSQL databases is typically stored in a way that is optimized for queries. The rule of thumb when you use MongoDB is data that is accessed together should be stored together. Queries typically do not require joins, so the queries are very fast.

Easy for developers: Some NoSQL databases like MongoDB map their data structures to those of popular programming languages. This mapping allows developers to store their data in the same way that they use it in their application code. While it may seem like a trivial advantage, this mapping can allow developers to write less code, leading to faster development time and fewer bugs.

Q2 What are business drivers for NoSQL?
Business drivers apply pressure to single CPU system resulting in the cracks

- Volume and Velocity refer to ability to handle big data
- Variability refers to how diverse data types don't fit into structured tables
- Agility refers to how quickly an organization responds to business change

Q3 Data Architecture Patterns?
Key-Value Store Database: This model is one of the most basic models of NoSQL databases. As the name suggests, the data is stored in form of Key-Value Pairs. The key is usually a sequence of strings, integers or characters but can also be a more advanced data type. The value is typically linked or co-related to the key. The key-value pair storage databases generally store data as a hash table where each key is unique. The value can be of any type (JSON, BLOB(Binary Large Object), strings, etc). This type of pattern is usually used in shopping websites or e-commerce applications.

- **Advantages:**
  - Can handle large amounts of data and heavy load,
  - Easy retrieval of data by keys.

- **Limitations:**
  - Complex queries may attempt to involve multiple key-value pairs which may delay performance.
  - Data can be involving many-to-many relationships which may collide.

Examples: DynamoDB

Column Store Database:
Rather than storing data in relational tuples, the data is stored in individual cells which are further grouped into columns. Column-oriented databases work only on columns. They store large amounts of data into columns together. Format and titles of the columns can diverge from one row to other. Every column is treated separately. But still, each individual column may

contain multiple other columns like traditional databases. Basically, columns are mode of storage in this type.

Advantages:

Data is readily available

Queries like SUM, AVERAGE, COUNT can be easily performed on columns.

Examples:

HBase

Bigtable by Google

Cassandra

Document Database:

The document database fetches and accumulates data in form of key-value pairs but here, the values are called as Documents. Document can be stated as a complex data structure. Document here can be a form of text, arrays, strings, JSON, XML or any such format. The use of nested documents is also very common. It is very effective as most of the data created is usually in form of JSONs and is unstructured.

Advantages:

This type of format is very useful and apt for semi-structured data.

Storage retrieval and managing of documents is easy.

Limitations:

Handling multiple documents is challenging

Aggregation operations may not work accurately.

Examples: MongoDB

Graph Databases:

Clearly, this architecture pattern deals with the storage and management of data in graphs. Graphs are basically structures that depict connections between two or more objects in some data. The objects or entities are called as nodes and are joined together by relationships called Edges. Each edge has a unique identifier. Each node serves as a point of contact for the graph. This pattern is very commonly used in social networks where there are a large number of entities and each entity has one or many characteristics which are connected by edges. The relational database pattern has tables that are loosely connected, whereas graphs are often very strong and rigid in nature.

Advantages:

Fastest traversal because of connections.

Spatial data can be easily handled.

Limitations: Wrong connections may lead to infinite loops.

Examples: Neo4J

Pattern name	Description	Typical uses
Key-value store	A simple way to associate a large data file with a simple text	Dictionary, image store, document/file store, query cache, lookup tables
Column family (Bigtable) store	A way to store sparse matrix data using a row and a column as the key	Web crawling, large sparsely populated tables, highly-adaptable systems, systems that have high variance
Document store	A way to store tree-structured hierarchical information in a single unit	Any data that has a natural container structure including office documents, sales orders, invoices, product descriptions, forms, and web pages; popular in publishing, document exchange, and document search

Q4 What do you understand by BASE properties of NoSQL?

BASE stands for:

- **Basically Available** – Rather than enforcing immediate consistency, BASE-modelled NoSQL databases will ensure availability of data by spreading and replicating it across the nodes of the database cluster.
- **Soft State** – Due to the lack of immediate consistency, data values may change over time. The BASE model breaks off with the concept of a database which enforces its own consistency, delegating that responsibility to Developers.
- **Eventually Consistent** – The fact that BASE does not enforce immediate consistency does not mean that it never achieves it. However, until it does, data reads are still possible (even though they might not reflect the reality).

Module 4

Q1 Explain Data stream management Architecture?

1. **Input streams :** The input streams have the following characteristics:_
   - There can be one or more number of input streams entering the system.
   - The streams can have different data types
   - The rate of data flow of each stream may be different.
   - Within a stream the time interval between the arrival of data items may differ. For example, suppose the second item arrives after 2 ms from the arrival of the first data item, then it is not necessary that the third data item arrive after 2 ms from the arrival of the second data item. It may arrive earlier or even later.

2. **Stream processor:** : All types of processing such as sampling, cleaning, filtering, and querying on the input stream data are done here. Two types of queries are supported which are standing queries and ad-hoc queries.

3. Working: A limited memory such as a disk or main memory is used as the working storage for storing parts of summaries stories so that queries can be executed. If faster processing is needed, main memory is used, otherwise a secondary storage disk is used. As the working storage is limited in size, it is not possible to store all the data received from all the streams.

4. Archival: : The archival store is a large· storage area· in which the streams may be archived but execution of queries directly on the archival store is not supported. Also, the fetching of data from this store takes a lot of time as The archival store is a large· storage area· in which the streams may be archived but execution of compared to the fetching of data from the working store.

5. Outpμt streams : The output consists of the fully processed streams and the results of the execution of queries on the streams.

The difference between a conventional database-management system and a data-stream management system is that in case of the database-management system all of the data is available on the disk and the system can control the rate of data reads. On the other hand, in case of the data-stream-management system the rate of arrival of data is not in the control of the system and the system has to take care of the possibilities of data getting lost and take the necessary precautionary measures.

Q2 What are the types of stream? Give example

Transactional data stream –
It is a log interconnection between entities

- Credit card – purchases by consumers from producer
- Telecommunications – phone calls by callers to the dialed parties
- Web – accesses by clients of information at servers

Measurement data streams –

- Sensor Networks – a physical natural phenomenon, road traffic
- IP Network – traffic at router interfaces
- Earth climate – temperature, humidity level at weather stations

Sources:
Sensor Data –
In navigation systems, sensor data is used. Imagine a temperature sensor floating about in the ocean, sending back to the base station a reading of the surface temperature each hour. The data generated by this sensor is a stream of real numbers. We have 3.5 terabytes arriving every day and we for sure need to think about what we can be kept continuing and what can only be archived.

Image Data –
Satellites frequently send down-to-earth streams containing many terabytes of images per day. Surveillance cameras generate images with lower resolution than satellites, but there can be numerous of them, each producing a stream of images at a break of 1 second each.

Internet and Web Traffic –
A bobbing node in the center of the internet receives streams of IP packets from many inputs and paths them to its outputs. Websites receive streams of heterogeneous types. For example, Google receives a hundred million search queries per day.

Q3 What is DGIM algorithm? OR Which algorithm is used for Counting Ones in a Window ?

The DGIM Algorithm (Datar - Gionis - Indyk - Motwani)

The simplest case of the algorithm uses O(log2 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%. Maximum number of errors in estimation of the number 1's in the window is 50%

The two basic components of this algorithm are :

- o Timestamps: The sequence nümber of in the order of arrival
- o Buckets.

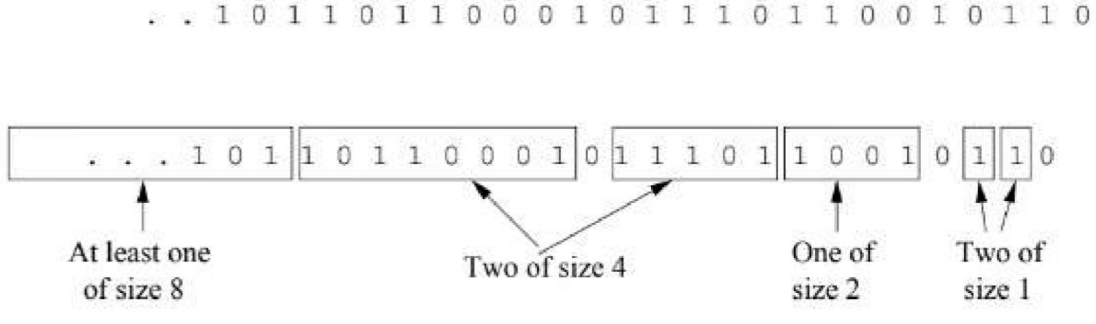
Each bit arriving in the stream is assigned a timestamp. So, the first bit is assigned timestamp 1, the second bit is assigned timestamp 2 and each subsequent bit is assigned a timestamp one higher than the previous bit.

We divide the window into buckets, 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 the bucket.

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 a 1.
- Every position with a 1 is in some bucket.
- No position is in more than one bucket.
- There are one or two buckets of any given size, up to some maximum size.
- All sizes must be a power of 2.
- Buckets cannot decrease in size as we move to the left (back in time).

Example of counting ones

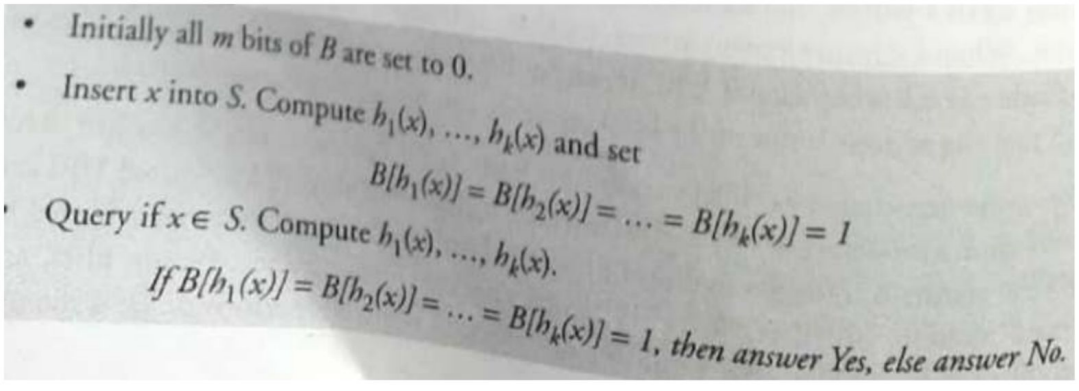


Q4 Define Bloom's filter.

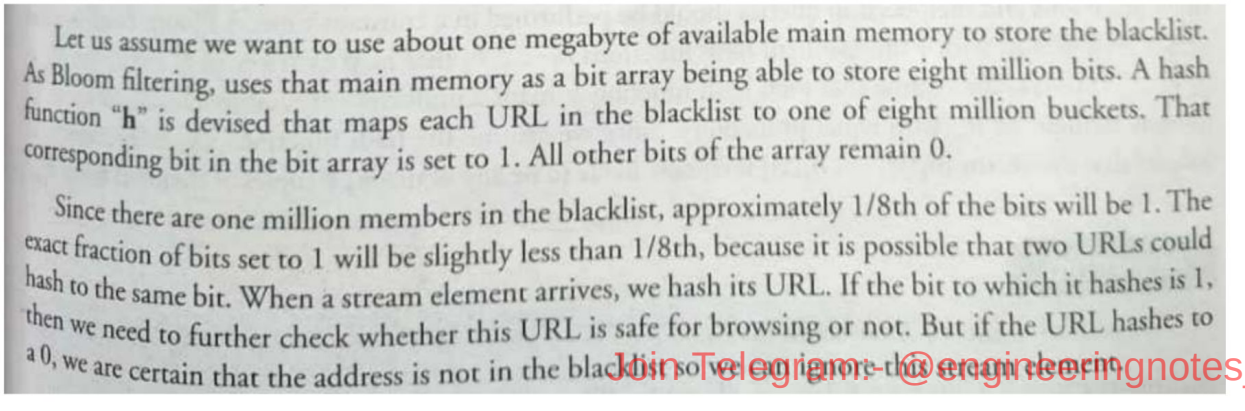
- An array of n bit; initialized to 0's. . . s a key to one of the n bit locations :
- A set H of k hash functions each of which map h1, h2, ... , hk
- A set S consisting of m number of keys.

1. Bloom filter accepts all those tuples whose keys are members of the set S and rejects all other tuples.
2. Initially all the bit locations in the array are set to 0. Each key K is taken from S and one by one all Of the k hash functions in H are applied on this key K. All bit locations produced by hi(K) are set to 1. hl(K), ...i X(K)
3. On arrival of a test tuple with key K, all the k different hash functions from H are once again applied on the test key K. If all the bit locations thus produced are 1, the tuple with the key K is accepted. If even a single bit location out of these turns out to be 0, then the tuple is rejected.
4. A Bloom filters is like a hash table ,and simply uses one bit to keep track whether an item hashed to the location.
5. If k=1 , it's equivalent to a hashing based fingerprint system.
6. If n=cm for small constant c,such as c=8 ,then k=5 or 6 ,the false positive probability is just over 2% .
7. It's interesting that when k is optimal k=ln(2)*(n/m) , then p= 1/2. An optimized Bloom filters looks like a random bit-string

Algo:



Example


Let us assume we want to use about one megabyte of available main memory to store the blacklist. As Bloom filtering, uses that main memory as a bit array being able to store eight million bits. A hash function "**h**" is devised that maps each URL in the blacklist to one of eight million buckets. That corresponding bit in the bit array is set to 1. All other bits of the array remain 0.

Since there are one million members in the blacklist, approximately 1/8th of the bits will be 1. The exact fraction of bits set to 1 will be slightly less than 1/8th, because it is possible that two URLs could hash to the same bit. When a stream element arrives, we hash its URL. If the bit to which it hashes is 1, then we need to further check whether this URL is safe for browsing or not. But if the URL hashes to a 0, we are certain that the address is not in the blacklist so we can ignore this stream element.

Q5 What do you mean by counting Distinct Elements in a stream. Illustrate with an example working of a Flajolet-Martin Algorithm used to count number of distinct elements. Which algorithm is used to count frequent elements

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.

Flajolet-Martin Algorithm is used for estimating the number of unique elements in a stream in a single pass. The time Complexity of this algorithm is O(n) and the space complexity is O(log m) where n is the number of elements in the stream and m is the number of unique elements. Its idea is that the more different elements we see in the stream, the more different hash-values we shall see. 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.

The major components of this algorithm are :

- o A collection of hash functions, and
- o A bit-string of length L such that n. A 64-bit string is sufficient for most cases.

Each incoming element will be hashed using all the hash functions. Higher the number Of distinct elements in the stream, higher will be the number of different hash values.

On applying a hash function h on an element of the stream e, the hash value h(e) is produced. We convert h(e) into an equivalent binary bit-string. This bit string will end in some number of zeroes. For instance, the 5-bit string 11010 with 1 zero and 10001 ends with no zeroes.

This count of zeroes is known as the tail length. If R denotes the maximum tail length of any element e encountered thus far in the stream, then the estimate for the number of unique elements in the stream is 2^R.

Now to see that this estimate makes sense we have to use the following arguments using probability theory:

- o The probability of h(e) having a tail length of at least r is 2^{-r}.
- o The probability that none of the m distinct elements have tail length of at least r is $(1-2^{-r})^m$.
- o The above expression can also be written as .$((1-2^{-r})^{2r})^{m2-r}$
- o And finally, we can reduce it to e^{-m2-r} as $(1-2-r)^{2r}= e^{-1}$

- If m » 2~ the probability of finding a tail of length at least r approaches 1 .
- If m « 2', the probability of finding a tail of length at least r approaches o.

So, we can conclude that the estimate of 2R is neither going to be too low nor too high.

- Let us now understand the working of the algorithm with an example :

Stream: 5, 3, 9, 2, 7, 11

)n :

$$h(x) = 3x + 1 \bmod 32$$

$$h(5) = 3(5) + 1 \bmod 32 = 16 \bmod 32 = 16 = 10000$$

$$h(3) = 3(3) + 1 \bmod 32 = 10 \bmod 32 = 10 = 01010$$

$$h(9) = 3(9) + 1 \bmod 32 = 28 \bmod 32 = 28 = 11100$$

$$h(2) = 3(2) + 1 \bmod 32 = 7 \bmod 32 = 7 = 00111$$

$$h(7) = 3(7) + 1 \bmod 32 = 22 \bmod 32 = 22 = 10110$$

$$h(11) = 3(11) + 1 \bmod 32 = 34 \bmod 32 = 2 = 00010$$

Tail lengths: {4, 1, 2, 0, 1, 1}

$$R = \max(\text{Tail length}) = 4$$

$$\text{Estimate of m} = 2^R = 2^4 = 16.$$

stimate is obt

Q6 What are challenges when querying on large data set?

- The data model and query semantics must allow order-based and time-based operations (e.g. queries over a five-minute moving window).
- The inability to store a complete stream suggests the use of approximate summary structures, As a result, queries over the summaries may not return exact answers.
- Streaming query plans may not use blocking operators that must consume the entire input before any results are produced.
- Due to performance and storage constraints, backtracking over a data stream is not feasible. On-line stream algorithms are restricted to making only one pass over the data.
- Applications that monitor streams in real-time must react quickly to unusual data values.
- Long-running queries may encounter changes in system conditions throughout their execution lifetimes (e.g. variable stream rates).
- Shared execution of many continuous queries is needed to ensure scalability

Module 5

Q1 Define Recommender system. And their types/ Explain Content-based recommendation Explain Collaborative filtering.

A recommendation system is an artificial intelligence or AI algorithm, usually associated with machine learning, that uses Big Data to suggest or recommend additional products to consumers. These can be based on various criteria, including past purchases, search history, demographic information, and other factors. Recommender systems are highly useful as they help users discover products and services they might otherwise have not found on their own.

Recommender systems are trained to understand the preferences, previous decisions, and characteristics of people and products using data gathered about their interactions. These include impressions, clicks, likes, and purchases. Because of their capability to predict consumer interests and desires on a highly personalized level, recommender systems are a favorite with content and product providers. They can drive consumers to just about any product or service that interests them, from books to videos to health classes to clothing.

Types/ What is the category for recommendation of List of favourites? What is the Category for recommendation of Top 10, Most Popular videos on YouTube? Explain Hybrid recommendation

Collaborative filtering algorithms recommend items (this is the filtering part) based on preference information from many users (this is the collaborative part). This approach uses similarity of user preference behavior, given previous interactions between users and items, recommender algorithms learn to predict future interaction. These recommender systems build a model from a user's past behavior, such as items purchased previously or ratings given to those items and similar decisions by other users. The idea is that if some people have made similar decisions and purchases in the past, like a movie choice, then there is a high probability they will agree on additional future selections. For example, if a collaborative filtering recommender knows you and another user share similar tastes in movies, it might recommend a movie to you that it knows this other user already likes.

Content filtering, by contrast, uses the attributes or features of an item (this is the content part) to recommend other items similar to the user's preferences. This approach is based on similarity of item and user features, given information about a user and items they have interacted with (e.g. a user's age, the category of a restaurant's cuisine, the average review for a movie), model the likelihood of a new interaction. For example, if a content filtering recommender sees you liked the movies You've Got Mail and Sleepless in Seattle, it might recommend another movie to you with the same genres and/or cast such as Joe Versus the Volcano.

Hybrid recommender systems combine the advantages of the types above to create a more comprehensive recommending system

Q2 What is Utility Matrix

The goal of our recommendation system is to build an mxn matrix (called the utility matrix) which consists of the rating (or preference) for each user-item pair. Initially, this matrix is usually very sparse because we only have ratings for a limited number of user-item pairs.

Q3 What is TF-IDF?

TF-IDF, short for Term Frequency - Inverse Document Frequency, is a text mining technique, that gives a numeric statistic as to how important a word is to a document in a collection or corpus. This is a technique used to categorize documents according to certain words and their importance to the document. This is somewhat similar to Bag of Words (BoW), where BoW is an algorithm that counts how many times a word appears in a document. If a perticular term appears in a document, many times, then there is a possibility of that term being an important word in that document. This is the basic concept of BoW, where the word count allows us to compare and rank documents based on their similarities for applications like search, document classification and text mining. Each of these are modeled into a vector space so as to easily categorize terms and their documents. But the general BoW technique, does not omit the common words that appear in documents and it is also being modeled in the vector space.

TF-IDF allows us to score the importance of words in a document, based on how frequently they appear on multiple documents.

1. If the word appears frequently in a document - assign a high score to that word (term frequency - TF)
2. If the word appears in a lot of documents - assign a low score to that word. (inverse document frequency - IDF)

Calculating Term Frequency

$$\text{tf}(t, d) = \frac{Frequency\ of\ term\ t, in\ document\ d}{Total\ number\ of\ terms\ in\ document\ d}$$

Calculating Inverse Document Frequency

$$\text{idf}(t) = log\ \frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ term\ t\ in\ it}$$

$$\textbf{tf-idf}_{(t,d)} = \textbf{tf}_{(t,d)} \textbf{ x idf}_{(t)}$$

t = term

d = document

Q4 What is the Cold-Start problem? What is User Cold Start Problem and Item/Product Cold Start Problem? Ways to mitigate/solve them

Recommender systems are a sort of information filtering technology that aims to offer information items that are likely to be of interest to the user. The cold start problem occurs when the system is unable to form any relation between users and items for which it has insufficient data. There are two types of cold-start problems: –

- **User cold-start problems**: When there is almost no information available about the user, the user cold-start problem arises.
- **Product cold-start problems**: When there is almost no information about the product, the product cold-start problem arises.

Mitigation Techniques

Representative Approach: If we lack sufficient knowledge of users and objects, we can rely on those who represent the set of items and users. That is the underlying idea of representative-based approaches. Representatives can be users whose linear combinations of preferences closely resemble those of other users.

Feature Mapping: The matrix factorization technique may be used to solve feature mapping. The basic concept is that a matrix factorization model depicts user-item interactions as the product of two rectangular matrices whose content is learnt using known interactions via machine learning. Each user will be assigned a row in the first matrix, and each item will be assigned a column in the second matrix; this set is called a latent factor. Latent factors are rows or columns that are linked with a certain individual or object.

Hybrid Approach: Another new technique that is related to feature weighting is the development of a hybrid content-based filtering recommender in which characteristics, either of the items or of the people, are weighted depending on the user's sense of significance. Different criteria (such as the actors, director, region, and title) will be weighted differently in order to select a movie that the consumer would enjoy.

Q5 What is Sparsity

A major problem limiting the usefulness of collaborative filtering is the sparsity problem, which refers to a situation in which transactional or feedback data is sparse and insufficient to identify similarities in consumer interests. when recommending from a large item set, users will have rated only some of the items (makes it hard to find similar users)

Q6 Explain Girvan-Newman algorithm to mine Social Graphs.
Divisive method: starts with the full graph and breaks it up to find communities.
Too slow for many large networks (unless they are very sparse), and it tends to give relatively poor results for dense networks.
The edge betweenness for edge e:

$$\sum_{s,t \neq v} \frac{\sigma_{st}(e)}{\sigma_{st}},$$

Where σst is total number of shortest paths from node s to node t and σst(e) is the number of those paths that pass through e.
The summation is n(n − 1) pairs for directed graphs and n(n − 1)/2 for undirected graphs.

Steps(BFS -> DAG -> Recalculate ->Split/Graph Partitioning)
Step 1. Find the edge of highest betweenness - or multiple edges of highest betweenness, if there is a tie - and remove these edges from the graph. This may cause the graph to separate into multiple components. If so, this is the first level of regions in the partitioning of the graph.

Step 2. Recalculate all betweennesses, and again remove the edge or edges of highest betweenness. This may break some of the existing components into smaller components; if so, these are regions nested within the larger regions.

Step 3 Proceed in this way as long as edges remain in graph, in each step recalculating all betweennesses and removing the edge or edges of highest betweenness

Q7 Give Applications of Social Network mining.
A social graph is a diagram that illustrates interconnections among people, groups and organizations in a social network. The term is also used to describe an individual's social network. When portrayed as a map, a social graph appears as a set of network nodes that are connected by lines

Applications
Social media applications
information networks (documents, web graphs, patents),
infrastructure networks (roads, planes, water pipes, powergrids),
biological networks (genes, proteins, food-webs of animals eating each other),
product co-purchasing networks(e.g., Groupon).

Q8 How is classification algorithm used in recommendation system
It is used to know the user's interest. By applying some function to new item we get . • some probability which user may like.Numeric values also help to know about the degree of interest with some particular item.

Few of techniques are listed as follows :
(1) Decision Tree and Rule Induction

(2) Nearest Neighbour Method
(3) Euclidean Distance Metric
(4) Cosine similarity functuin
Some other classification algorithm are
1) Relevance feedback and Rocchio's al · . h
2) Linear classification
3) probabilistic methods
4) Naive Bayes.