USN											

Internal Assessment Test 2 – NOV/DEC 2025
Scheme of Evaluation

Sub:	Big Data Analytics				Sub Code:	BIS701	Branch:	ISE
Date:	02/12/2025	Duration:	90 min	Max Marks: 50	Sem/Sec:	VII / A, B & C		OBE

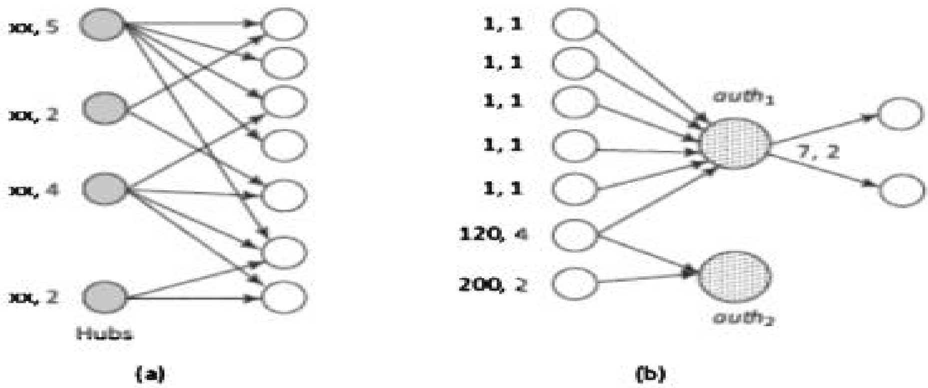
	Answer any FIVE FULL Questions	MARKS	CO	RBT
1	Discuss the HITS algorithm used in web mining. Explain the iterative process required to calculate authority and hub weights.	[10]	4	L2

Scheme : Explanation+Diagram+Algorithm carries 3+3+4 marks

Solution :

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

Figure 9.10(a) shows hubs (shaded circles) with the number of out-links associated with each hub. Figure 9.10(b) shows authorities (dotted circles) with the number of in-links and out-links associated with each link.



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

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

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

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

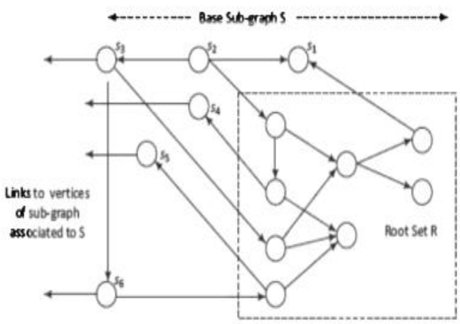
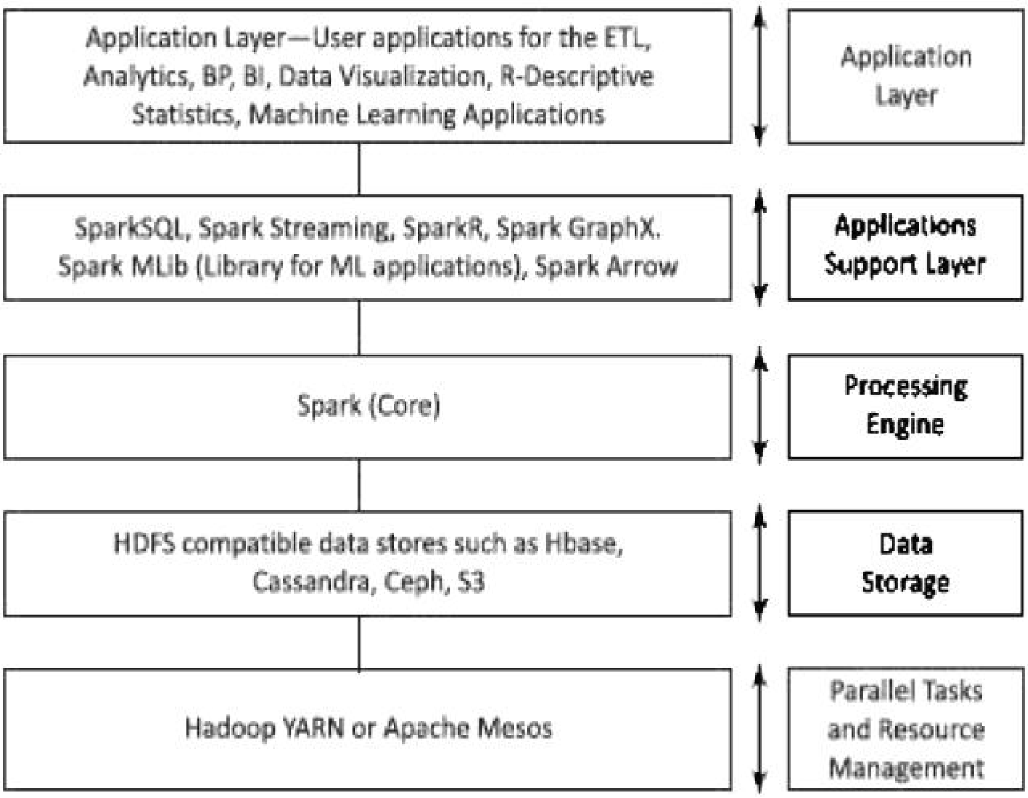


Figure 9.11 Sub-graph for HITS consisting of root set R of pages and base sub-graph **S** including all the pages pointed to by any page of R.

2	Explain the five-layer architecture used for running applications in the Apache Spark stack. **Scheme :** Explanation+Diagram carries 3+7 marks **Solution :**	[10]	3	L2



Spark Stack

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

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

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

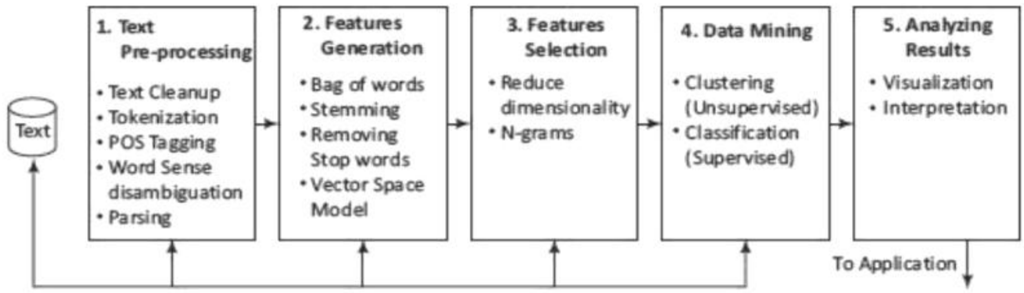
SparkR is an R package used as light-weight front end for Apache Spark from R. Spark API uses SparkR through the RDD class. A user can interactively run the jobs from the R shell on a cluster. An RDD API is in the distributed lists in R.

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

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

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

3	Explain the sequential stages of a typical text mining pipeline. Scheme : Diagram + Explanation of each phase carries 2+8 marks Solution:	[10]	4	L2



1. Text Pre-processing

- **Cleans and prepares raw text** by removing noise (HTML tags, punctuation, extra spaces).
- **Tokenizes text** into words or sentences and applies POS tagging to identify grammatical roles.
- **Resolves ambiguities** (like word sense) and performs parsing to understand sentence structure.

2. Feature Generation

- Converts text into **numeric or vector formats** such as Bag-of-Words or vector space models.
- Applies **stemming/lemmatization** to reduce words to their root form.
- Removes **stop words** and irrelevant terms to improve model accuracy.

3. Feature Selection

- **Reduces dimensionality** by selecting the most meaningful features.
- Uses **N-grams** to capture context (word sequences).
- Helps improve model performance by removing noisy or redundant features.

4. Data Mining

- **Clustering** is applied to group similar documents without labels (unsupervised).
- **Classification** assigns documents into predefined categories (supervised).
- Uses machine learning techniques to extract hidden patterns from text data.

5. Analyzing Results

- Results are **visualized** using charts, graphs, or word clouds to understand patterns.
- **Interpretation** helps in decision-making and understanding model insights.
- Output is fed into an application for real-world use (e.g., recommendation, sentiment analysis).

4	**Describe the features, data units and aggregate methods in Hive with examples.** Scheme : HIVE features, data units and aggregate methods carries 3+3+4 marks Solution:	[10]	4	L2

1. Hive Features

Feature	Description	Example / Explanation
SQL-like Language (HiveQL)	Hive provides SQL-like queries, easy for SQL users.	SELECT name, salary FROM employees WHER salary > 50000;
Schema-on-Read	Schema applied when reading data, not when loading. Allows flexible data formats.	Loading raw CSV/JSON data without validation.
Works on Hadoop (HDFS + YARN)	Uses Hadoop for storage and execution (MapReduce/Tez/Spark).	Query on TB/PB of logs stored in HDFS.
Handles Large Data (Petabyte scale)	Suitable for batch processing of huge datasets.	Analyzing clickstream log of billions of records.
Supports Partitioning	Divides table data into folders based on key columns → faster queries.	... PARTITIONED BY (year, month);
Supports Bucketing	Further divides partitions based on hash of a column.	CLUSTERED BY (id) INTO 4 BUCKETS;

2. Hive Data Units

Data Unit	Definition	Purpose	Example
Database	Logical namespace containing tables.	Organizes tables by domain or project.	CREATE DATABASE retail; USE retail;
Table	Structured data stored in rows & columns.	Main data storage object in Hive.	CREATE TABLE employees(INT, name STRING);
Partition	Subdivision of table data based on column values; stored as directory.	Improves query speed by scanning only relevant partitions.	Folder: /sales/year=2024/month=01/
Bucket	Hash-based subdivisions of partitions.	Efficient for sampling & joins.	CLUSTERED BY (id) INTO 4 BUCKETS;
Row	A record inside a table.	Represents a single data entry.	{ id: 1, name: 'John' }
Column	Fields inside each row.	Holds typed attributes.	salary FLOAT
Metastore	Stores metadata: table structure, partitions, columns.	Used by Hive for query planning.	Backend DB like MySQL/PostgreSQL.

3. Hive Aggregate Methods

Function	Description	Example Query
COUNT()	Counts number of rows.	SELECT COUNT(*) FROM employees;
SUM()	Adds numeric values.	SELECT SUM(amount) FROM sales;
AVG()	Calculates average.	SELECT AVG(salary) FROM employees;
MIN()	Finds smallest value.	SELECT MIN(price) FROM products;
MAX()	Finds largest value.	SELECT MAX(score) FROM students;
STDDEV()	Standard deviation.	SELECT STDDEV(salary) FROM employees;
VARIANCE()	Variance.	SELECT VARIANCE(age) FROM people;

5.a Given two datasets: **Orders:** (order_id:int, product_id:int, quantity:int) [6] 3 L3

Products: (product_id:int, product_name:chararray, price:float)

Write a Pig Latin script to:

1. Join the datasets to attach product names and prices.
2. Calculate the **revenue generated per product**.
3. Sort the products by revenue in descending order.

Scheme : Script writing for each sub question carries 2 marks each

Solution :

```
-- Load the Orders dataset
orders = LOAD 'orders'
   USING PigStorage(',')
   AS (order_id:int, product_id:int, quantity:int);

-- Load the Products dataset
products = LOAD 'products'
   USING PigStorage(',')
   AS (product_id:int, product_name:chararray, price:float);

1. Join datasets on product_id
joined_data = JOIN orders BY product_id, products BY product_id;

2. Calculate revenue per product
Revenue = quantity * price
revenue_calc = FOREACH joined_data GENERATE
   products::product_id AS product_id,
   products::product_name AS product_name,
   (quantity * price) AS revenue;
```

Group by product to sum revenue across all orders

```
revenue_grouped = GROUP revenue_calc BY (product_id, product_name);

revenue_per_product = FOREACH revenue_grouped GENERATE
    group.product_id AS product_id,
    group.product_name AS product_name,
    SUM(revenue_calc.revenue) AS total_revenue;
```

3. Sort by revenue in descending order

```
sorted_products = ORDER revenue_per_product BY total_revenue DESC;
```

Store or dump result

```
DUMP sorted_products;
```

5.b A sales table has the schema: (sale_id INT, product STRING, category STRING, quantity INT, amount DOUBLE).Write a **Hive** query to find the **total sales amount** for each category and display only categories with total sales **greater than 50,000**. [4]

Scheme : Query writing for each sub question carries 2 marks each

Solution :

```
SELECT
    category,
    SUM(amount) AS total_sales
FROM sales
GROUP BY category
HAVING SUM(amount) > 50000;
```

---

6 **How are arrays represented and manipulated in MongoDB? Explain with a proper example.** [10] 3 L2

Scheme : Explanation of Array Representation + Manipulation with examples carries 5 marks each.

Solution :

In **MongoDB**, arrays are a native data type and can store multiple values of any type, including nested documents and other arrays.

Arrays are written directly inside documents using square brackets [].

```
{
  "_id": 1,
  "name": "Alice",
  "skills": ["MongoDB", "Python", "Machine Learning"],
  "scores": [85, 92, 88]
}
```

Common Array Manipulations:-

1. Insert a Document Containing an Array

```
db.users.insertOne({
  _id: 1,
  name: "Alice",
  hobbies: ["Reading", "Traveling", "Music"],
  scores: [85, 92, 88],
  addresses: [
    { type: "home", city: "New York" },
    { type: "office", city: "Boston" }
  ]
});
```

2. Query Documents Based on Array Values

```
db.users.find({ hobbies: "Cycling" });
```

3. Add Elements to an Array

```
db.users.updateOne(
  { name: "Bob" },
  { $push: { hobbies: "Swimming" } }
);
```

4. Remove Elements From an Array

```
db.users.updateOne(
  { name: "Bob" },
  { $pull: { hobbies: "Cycling" } }
);
```

5. Prevent Duplicates Using $addToSet

```
db.users.updateOne(
  { name: "Bob" },
  { $addToSet: { hobbies: "Reading" } }
);
```

6. Update Elements in an Array

```
{
  "_id": 1,
  "scores": [85, 92, 88]
}
```

Faculty Signature CCI Signature HOD Signature