USN | | | | | | | | | | |

Internal Assessment Test 2– NOV-DEC 2025

Sub:	BIG DATA ANALYTICS -Schemeand solution				Sub Code:	BCS714D	Branch:	CSE
Date:	29/11/2025	90 mins	Max Marks:	50	Sem/Sec:	VII/ A/B/C		OBE

Answer any FIVE FULL Questions MARKS CO RBT

		MARKS	CO	RBT
1	Analyze different phases of text mining process with a neat diagram	10	CO1	L2

Diagram -5,Explaination-5

Text mining (also called Text Analytics) is the process of converting unstructured text into meaningful and actionable insights. It typically consists of the following phases:

---

1. Text Collection (Data Gathering)

- First step where raw text data is collected.
- Sources: social media posts, emails, documents, PDFs, web pages, research papers, logs, reviews, etc.
- Purpose: gather sufficient, relevant textual data for analysis.

---

2. Text Pre-processing (Cleaning)

This is the most important stage to improve data quality. It includes:

- **Tokenization** – breaking sentences into words/tokens.
- **Stop-word Removal** – removing common words like *is, the, a*.
- **Stemming / Lemmatization** – converting words to root form (e.g., “running” → “run”).
- **Lowercasing** – converting all text to lowercase.
- **Removing punctuation, numbers, emojis**, etc.

This step converts unstructured text into clean, processable data.

---

3. Text Transformation (Representation)

Transforms cleaned text into numerical or structured form.

Common methods:

- **Bag of Words (BoW)**
- **TF–IDF (Term Frequency–Inverse Document Frequency)**
- **N-grams**
- **Word Embeddings (Word2Vec, GloVe)**
- **Sentence Embeddings (BERT, SBERT)**

The output is a feature matrix usable by ML algorithms.

---

4. Pattern Discovery (Mining / Analysis)

Actual text mining happens here. Methods include:

- **Classification** (spam detection, sentiment classification)
- **Clustering** (grouping similar documents)
- **Topic Modeling** (LDA)
- **Information Extraction**
- **Association Rule Mining**
- **Sentiment Analysis**

This step finds hidden patterns, relationships, or insights.

---

5. Evaluation & Interpretation

- Assess quality of results using metrics like accuracy, precision, recall, F1-score for classification.
- Check coherence of topics for topic modeling.
- Interpret mined patterns to gain insights.

---

6. Knowledge Presentation

- Present results visually or textually:
    - Graphs
    - Word clouds
    - Dashboards
    - Summaries

- Final output used for decision-making.

Neat Diagram of Text Mining Process

```
 +----------------------+
 |   Text Collection    |
 |  (Raw Data Sources)  |
 +----------+-----------+
            |
            v
 +----------------------+
 |  Text Pre-processing |
 |  (Cleaning & Normal) |
 +----------+-----------+
            |
            v
 +----------------------+
 |  Text Transformation |
 | (Feature Extraction) |
 +----------+-----------+
            |
            v
 +----------------------+
 |   Pattern Discovery  |
 | (ML / NLP Techniques)|
 +----------+-----------+
            |
            v
 +----------------------+
 |  Evaluation &        |
 |  Interpretation      |
 +----------+-----------+
            |
            v
 +----------------------+
 | Knowledge Presentation|
 | (Reports/Visualization)|
 +------------------------+
```

2	Consider the following text stored in HDFS:	10	CO2	L3

Data science helps discover patterns.
Machine learning supports data science methods.
Big data technologies fuel innovation.
Science and data together build intelligence.

Explain the different phases of the Map tasks and Reduce tasks involved in processing the above input for the Word Count problem

Steps--5,Explaination-5

.I. MAP TASK PHASES

Each line of text is processed by the Mapper. The Map Task has the following phases:

---

1. Input Splitting

- The input file is split into blocks (e.g., 128 MB).
- Each block is assigned to a mapper.
- Your input is small → likely one mapper processes all lines.

---

2. Record Reader

- Converts raw input into key-value pairs.
- Default: **key = byte offset**, **value = one line of text**.

Example output:

0: "Data science helps discover patterns."
52: "Machine learning supports data science methods."
...

---

3. Map Function

- The mapper takes each line (value), tokenizes it into words, and emits pairs: **(word, 1)**

Example mapper output for each line:

Line 1:
(data,1) (science,1) (helps,1) (discover,1) (patterns,1)

Line 2:
(machine,1) (learning,1) (supports,1)
(data,1) (science,1) (methods,1)

Line 3:
(big,1) (data,1) (technologies,1)
(fuel,1) (innovation,1)

Line 4:
(science,1) (and,1) (data,1)
(together,1) (build,1) (intelligence,1)

---

4. Combiner (Optional)

Acts like a **local mini-reducer** to reduce data transfer.

Example:
Mapper local output might combine duplicates:

(data,3)
(science,3)

---

5. Partitioning

- Determines which reducer receives which key.
- Default: **hash(word) mod number_of_reducers**.

---

6. Shuffle and Sort

- **Shuffle**: sends all values of each key to the correct reducer.
- **Sort**: sorts keys in alphabetical order before sending to reducers.

After sorting, reducer receives items like:

and → [1]
big → [1]
build → [1]
data → [1,1,1,1]
discover → [1]
fuel → [1]
helps → [1]
innovation → [1]
intelligence → [1]
learning → [1]
machine → [1]
methods → [1]
patterns → [1]
science → [1,1,1]
supports → [1]
technologies → [1]
together → [1]

II. REDUCE TASK PHASES

---

1. Shuffle and Merge

- Reducer receives sorted keys with list of values (from all mappers).

---

2. Reduce Function

- Sums up all the values for each key.

Example reducer output:

```
and → 1
big → 1
build → 1
data → 4
discover → 1
fuel → 1
helps → 1
innovation → 1
intelligence → 1
learning → 1
machine → 1
methods → 1
patterns → 1
science → 3
supports → 1
technologies → 1
together → 1
```

---

3. Final Output Writer

Reducer writes the final key-value pairs to HDFS:

```
and 1
big 1
build 1
data 4
discover 1
fuel 1
helps 1
innovation 1
intelligence 1
learning 1
machine 1
methods 1
patterns 1
science 3
supports 1
technologies 1
together 1
```

3	Explain GROUP and SPLIT relational operator in PIG with suitable example `GROUP` is used to **group the data based on one or more fields**. It works similar to SQL GROUP BY, but Pig returns a **bag** of tuples for each group. **5+5**	(5+5)	CO2	L2

	group_data = GROUP relation BY field;
	
	Input (students.txt)
	`John  A`
	`Mary  B`
	`Ravi  A`
	`Kiran C`
	`Rita  B`
	
	Pig Script
	`students = LOAD 'students.txt' USING PigStorage(' ')`
	`            AS (name:chararray, grade:chararray);`
	
	`grouped_data = GROUP students BY grade;`
	
	`DUMP grouped_data;`
	
	Output
	`(A,{(John,A),(Ravi,A)})`
	`(B,{(Mary,B),(Rita,B)})`
	`(C,{(Kiran,C)})`			

4	Examine identifiers, keywords , datatypes and operators of PIG with an example. **Identifiers-5,Datatypes-2.5,Operators-2.5** **1. Identifiers in Pig** **Definition** Identifiers are names given to: ● Relations ● Fields (column names) ● Aliases ● Variables **Rules** ● Must start with a **letter (A–Z or a–z)**. ● Can contain letters, digits, and underscore (_). ● Cannot start with a digit. ● Cannot use Pig **keywords** as identifiers. **Example** `students = LOAD 'std.txt' AS (name, age, dept);`	10	CO2	L2

Here:

- `students` → relation identifier
- `name`, `age`, `dept` → field identifiers (aliases)

---

2. Keywords in Pig

Definition

Keywords are **reserved words** in Pig Latin. They cannot be used as identifiers.

Common Pig Keywords

```
LOAD, STORE, FILTER, GROUP, FOREACH, ORDER, BY,
DISTINCT, JOIN, SPLIT, INTO, IF, AS, USING, DUMP
```

Example

```
emp = LOAD 'emp.txt' USING PigStorage(',') AS (id, name,
salary);
filtered = FILTER emp BY salary > 50000;
DUMP filtered;
```

- `LOAD`, `FILTER`, `BY`, `AS`, `DUMP` → keywords
- They instruct Pig on what operation to perform.

---

3. Datatypes in Pig

Pig supports both **simple** and **complex** datatypes.

A. Simple Datatypes

Datatype	Description	Example
`int`	32-bit integer	10
`long`	Large integer	9876543210
`float`	Single precision	12.5
`double`	Double precision	99.99
`chararray`	String	"Hello"

	`bytearray` Raw bytes default type **Example** `data = LOAD 'data.txt' AS (id:int, name:chararray, marks:float);`			

5.a	Define HIVE, Explain its main task and features.	5	CO2	L1

Define-5,Explaination-5

Hive is a **data warehousing and SQL-like querying system** built on top of **Hadoop**.
It allows users to **store, manage, and analyze large datasets** stored in HDFS using a language called **HiveQL**, which is similar to SQL.

Originally developed by **Facebook**, Hive makes big data processing easier for people who are comfortable with SQL instead of Java MapReduce.

The **main task** of Hive is:

🔹 **To convert SQL-like queries (HiveQL) into MapReduce, Tez, or Spark jobs for execution on Hadoop.**

In simple terms:

HiveQL → Execution Engine → Hadoop Jobs → Results

Hive handles:

- Query parsing
- Query optimization
- Job execution on Hadoop
- Managing data stored in HDFS

📌 Detailed Explanation of Hive's Main Tasks

1. **Data Storage Management**
    - Stores structured/semi-structured data in tables.
    - Metadata is stored in the **Hive Metastore**.
2. **Query Processing**
    - Converts high-level HiveQL queries into low-level jobs:
        - MapReduce
        - Tez

- Spark

3. **Schema Management**
    - Defines tables, partitions, and buckets.
    - Supports external and internal tables.
4. **Batch Processing**
    - Efficient for large-scale batch queries, aggregations, and summaries.
5. **Data Analysis**
    - Performs OLAP-style operations:
      GROUP BY, JOIN, ORDER, COUNT, MAX, MIN, etc.

Key Features of Hive

1. SQL-like Query Language — HiveQL

- Easy for users familiar with SQL.
- Supports SELECT, JOIN, GROUP BY, ORDER BY, etc.

2. Scalability

- Can process **petabytes** of data in HDFS.
- Scales horizontally across many nodes.

3. Extensibility

- Users can create:
    - **UDFs** (User Defined Functions)
    - **Custom SerDes** (Serializer/Deserializer)
    - **Custom formats**

4. Schema on Read

- Hive applies schema **when reading data**, not when storing.
- Makes Hive flexible for semi-structured data.

5. Batch Processing (NOT for OLTP)

- Suitable for data summarization, log analysis, reporting.

	● Not meant for real-time queries. **6. Supports Different Execution Engines** ● MapReduce (default, older) ● Apache Tez (faster) ● Apache Spark (fastest) **7. Supports Partitioning & Bucketing** ● Improves query performance by reducing scanned data. ● Partitioning: divides data based on column (e.g., date). ● Bucketing: divides partition into multiple files. **8. Fault Tolerant** ● Uses Hadoop’s fault tolerance and replication. **9. Metadata Stored in Metastore** ● Usually maintained in MySQL/PostgreSQL. ● Stores schema, table info, columns, partitions. **10. Compatible With Various File Formats** ● Text, CSV, ORC, Parquet, Avro, SequenceFile, JSON.			
5.b	Consider a Hive table named sales_data with the schema: sales_data(region STRING, product STRING, quantity INT, revenue DOUBLE) Write a Hive query to display all products whose total quantity sold is greater than 500 units. Your query should group the records by product and filter the grouped results using the HAVING clause. **Query-5Mark** SELECT product, SUM(quantity) AS total_quantity FROM sales_data GROUP BY product HAVING SUM(quantity) > 500;	5	CO2	L3
6..a	Identify and explain preprocessing steps, mining tasks for web content. **Steps-2.5,Explaination-2.5**	5	CO3	L2

I. Preprocessing Steps for Web Content

Web content comes from webpages containing text, HTML tags, scripts, multimedia, links, and advertisements.
To analyze such content, we must convert it into clean and structured data.

1. Content Extraction / HTML Parsing

- Removes HTML tags, JavaScript, CSS, ads, and navigation menus.
- Extracts useful text from <p>, <h1>, <title>, <a> tags.
- Tools: BeautifulSoup, JSoup, HTML parsers.

2. Tokenization

- Splits extracted text into words or tokens.
- Example:
  "Web mining is useful" → ["Web","mining","is","useful"]

3. Stop-word Removal

- Removes common but uninformative words such as:
  the, is, and, of, to, a, for
- Reduces noise in text mining.

4. Stemming / Lemmatization

- Converts words to root/base form.
- Example:
  running, runs, ran → run

5. Normalization

- Converts all text to lowercase.
- Removes punctuation, numbers, special symbols, emojis.
- Useful for reducing variations in similar words.

6. Feature Extraction / Vectorization

Converts text into numerical form for mining tasks.

Methods:

<table>
<tr><td></td><td>

Bag of Words (BoW)
TF–IDF
N-grams
Word embeddings (Word2Vec, GloVe)
Sentence embeddings (BERT)

<hr>
7. Noise Removal

Removes duplicates, boilerplate content, malformed HTML, scripts.

8. Representing Structured Web Data
For web pages with tables (HTML <table>):

Extract rows and columns.
Convert to structured formats like CSV or JSON.

</td><td></td><td></td><td></td></tr>
<tr><td>6.b</td><td>
Given the following two documents:

D1: Data science is exciting
D2: Data mining uses scientific methods

Calculate the TF–IDF score of the term “Data” in Document D1 using the appropriate formula.

Answer::-5

Step 1: Term Frequency (TF)

Document D1:
“Data science is exciting”

Total words in D1 = 4
Occurrences of “Data” in D1 = 1

TF=14=0.25TF = \frac{1}{4} = 0.25TF=41=0.25
<hr>
Step 2: Document Frequency (DF)

We check how many documents contain the term “Data”.
</td><td>5</td><td>CO3</td><td>L3</td></tr>
</table>

D1 → contains “Data”
D2 → “Data mining uses scientific methods” → contains “Data”

So,

DF=2DF = 2DF=2

Total number of documents:

N=2N = 2N=2

Step 3: Inverse Document Frequency (IDF)

IDF=log(NDF)=log(22)IDF = \log\left(\frac{N}{DF}\right) = \log\left(\frac{2}{2}\right)IDF=log(DFN)=log(22) IDF=log(1)=0IDF = \log(1) = 0IDF=log(1)=0

(In any log base, log(1) = 0.)

Step 4: TF–IDF Calculation

TF-IDF=TF×IDF=0.25×0=0TF\text{-}IDF = TF \times IDF = 0.25 \times 0 = 0TF-IDF=TF×IDF=0.25×0=0

CI CCI HoD