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

Internal Assessment Test 5 – Feb 2022 scheme and solution

Sub:	Big Data Analytics					Sub Code:	18CS72	Branch:	CSE
Date:	7/2/2022	Duration:	90 mins	Max Marks:	50	Sem/Sec:	VII/A,B,C		OBE

		MARKS	CO	RBT
1	Explain with diagram Pig architecture for scripts dataflow and processing. Pig architecture – 5 Dataflow and processing- 5	[10]	CO5	L2
2	Explain dump, for each, filter, joins, order by in Pig. for each, filter, joins, order by -2 marks each	[10]	CO5	L2
3	Write note on centralities, ranking and anomaly detection in social network analysis Centralities- 5 ranking and anomaly detection in social network analysis-5	[10]	CO5	L2
4	Explain PageRank algorithm. How to find TF and IDF? PageRank algorithm-5 TF and IDF- 5	[10]	CO4	L4
5	Explain web usage mining tasks. Detailed explanation – 10	[10]	CO5	L2

Apache PIG helps programmers write complex data transformations using scripts (without using Java).

- Pig Latin language is very similar to SQL and possess a rich set of built-in operators, such as group, join, filter, limit, order by, parallel, sort and split.
- It provides an interactive shell known as Grunt to write Pig Latin scripts.
- Programmers write scripts using Pig Latin to analyze data.
- The scripts are internally converted to Map and Reduce tasks with the help of the component known as Execution Engine, that accepts the Pig Latin scripts as input and converts these scripts into MapReduce jobs.
- Writing MapReduce tasks was the only way to process the data stored in HDFS before the Pig.

ii) Creates user defined functions (UDFs) to write custom functions which are not available in Pig. A UDF can be in other programming languages, such as Java, Python, Ruby, Jython,)Ruby. They easily embed into Pig scripts written in Pig Latin. UDFs provide extensibility to the Pig.

(iii) Process any kind of data, structured, semi-structured or unstructured data, coming from various sources.

(iv) Reduces the length of codes using multi-query approach. Pig code of 10 lines is equal to MapReduce code of 200 lines. Thus, the processing is very fast.

(v) Handles inconsistent schema in case of unstructured data as well.

(vi) Extracts the data, performs operations on that data and dumps the data in the required format in HDFS. The operation is called ETL (Extract Transform Load).

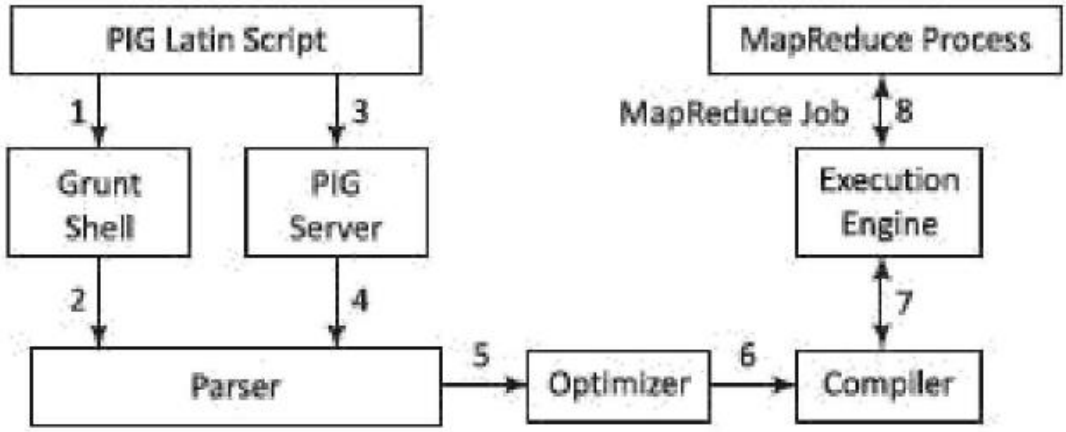
(vii) Performs automatic optimization of tasks before execution.

(viii) Programmers and developers can concentrate on the whole operation without a need to create mapper and reducer tasks separately.

(ix) Reads the input data files from HDFS or the data files from other sources such as local file system, stores the intermediate data and writes back the output in HDFS.

(x) Pig characteristics are data reading, processing, programming the UDFs in multiple languages and programming multiple queries by fewer codes. This causes fast processing.

(xi) Pig derives guidance from four philosophies, live anywhere, take anything, domestic and run as if flying. This justifies the name Pig, as the animal pig also has these characteristics.



The three ways to execute scripts are:

1. Grunt Shell: An interactive shell of Pig that executes the scripts.
2. Script File: Pig commands written in a script file that execute at Pig Server.
3. Embedded Script: Create UDFs for the functions unavailable in Pig built in operators. UDF can be in other programming languages. The UDFs can embed in Pig Latin Script file.

Parser : The Parser performs type checking and checks the script syntax.

The output is a Directed Acyclic Graph (DAG). Acylic means only one set of inputs are simultaneously at a node, and only one set of output generates after node operations.

DAG represents the Pig Latin statements and logical operators. Nodes represent the logical operators.

Edges between sequentially traversed nodes represent the dataflows.

Optimiser:

The DAG is submitted to the logical optimizer. The optimization activities, such as split, merge, transform and reorder operators execute in this phase.

The optimization is an automatic feature. The optimizer reduces the amount of data in the pipeline at any instant of time, while processing the extracted data.

It executes certain functions for carrying out this task, as explained as follows:

PushUpFilter: If there are multiple conditions in the filter and the filter can be split, **Pig splits the conditions and pushes up each condition separately.** Selecting these conditions at an early stage **helps in reducing the number of records** remaining in the pipeline.

PushDownForEachFlatten: Applying flatten, which produces a cross product between a complex type such as a tuple, bag or other fields in the record, as late as possible in the plan. This keeps **the number of records low in the pipeline**.

ColumnPruner: Omits never used columns or the ones no longer needed, reducing the size of the record. This can be applied after each operator, so that the fields can be pruned as aggressively as possible.

MapKeyPruner: Omitts **never used map keys**, reducing the size of the record.

LimitOptimizer: If the limit operator is immediately applied after *load* or *sort* operator, **Pig converts the load or sort into a limit-sensitive implementation,** which does not require processing the whole dataset. Applying the limit earlier reduces the number of records.

Compiler The compiler compiles after the optimization process. The optimized codes are a series of MapReduce jobs.

Execution Engine Finally, the MapReduce jobs submit for execution to the engine. The MapReduce jobs execute and it outputs the final result.

LOAD Command The first step to a dataflow is to specify the input. Load statement in Pig Latin loads the data from PigStorage.

To load data from HBase: book load 'MyBook' using HBaseStorage();

For reading CSV file, PigStorage takes an argument which indicates which character to use as a separator. For example, book = LOAD 'PigDemo/Data/Input/myBook.csv' USING PigStorage (,);

For reading text data line by line:

'PigDemo/Data/Input/myBook.txt' USING book = PigStorage() LOAD AS (lines: chararray);

To specify the data-schema for loading: book = LOAD 'MyBook' AS (name, author, edition, publisher);

Store Command

Pig provides the store statement for writing the processed data after the processing is complete. It is the **mirror image** of the load statement in certain ways.

By default, Pig stores data on HDFS in a tab-delimited file using **PigStorage:**

STORE processed into '/PigDemo/Data/Output/Processed';

To store in HBaseStorage with a *using* clause: STORE processed into 'processed' **using HBaseStorage();**

To store data as comma-separated text data, PigStorage takes an argument to indicate which character to use as a separator: STORE processed into 'processed' using PigStorage(',');

Dump Command

Pig provides dump command to see the processed data on the screen. This is particularly **useful during debugging and prototyping sessions. It can also be useful for quick adhoc jobs.**

The following command directs the output of the Pig script on the display screen:

DUMP processed;

Q2.

Foreach FOREACH gives a simple way to apply transformations based on columns. It is Pig's projection operator. Table 4.17 gives examples using FOREACH.

A = load 'input' as (name: chararray, rollno: long, address: chararray, phone: chararray, preferences: map [ ]) ;

Load an entire record, but then remove all but the name and phone fields from each record	B = foreach A generate name, phone; A= load 'input' as (t:tuple (x:int, y:int));
Tuple projection using dot operator	B = foreach A generate t.x, t.$1; A= load 'input' as (b:bag{t:(x:int, y:int)});
Bag projection	B = foreach A generate b.x; A= load 'input' as (b:bag{t:(x:int, y:int)});
Bag projection	B = foreach A generate b.(x, y); A= load 'input' as (x:chararray, y:int, z:int); Al = foreach A generate x, y + z as yz; B = group Al by x;
Add all integer values	C = foreach B generate SUM(Al.yz);

FILTER gives a simple way to select tuples from a relation based on some specified conditions (predicate). It is Pig's *select* command.

Loads an entire record, then selects the tuples with marks more than 75 from each record	A = load 'input' as (name:chararray, rollno:long, marks:float); B = filter A by marks > 75.0;
Find name (chararray) that do not match a regular expression by preceding the text without a given character string. Output is all names that do not start with P.	A= load 'input' as (name:chararray, rollno:long, marks:float); B = filter A by not name matches 'P.*';

ORDER statement sorts the data based on a specific field value, producing a total order of output data.

The syntax of order is similar to group. Key indicates by which the data sort.	A= load 'input' as (name: chararray, rollno: long, marks: float); B = order A by name;
To sort based on two or more keys (For example, first sort by, then sort by), indicate a set of keys by which the data sort. No parentheses around the keys when multiple keys indicate in order	A = load 'input' as (name:chararray,rollno:long, marks:float); B = order A by name, marks;

JOIN statement joins two or more relations based on values in the common field. Keys indicate the inputs. When those keys are equal, two tuples are joined. Tuples for which no match is found are dropped.

Join selects tuples from one input to put together with tuples from another input.	A = load 'input1' as (name:chararray, rollno:long); B = load 'input2' as (rollno:long, marks:float); C = join A by rollno, B by rollno
Also based on multiple keys join. All cases must have the same number of keys, and they must be of the same or compatible types.	A = load 'input1' as (name: chararray, fathername: chararray, rollno: long); B = load 'input2' as (name: chararray, rollno: long, marks: float); C = join A by (name, rollno), B by (name, rollno)

Pig also supports *outer joins*.

Tuples which do not have a match on the other side are included, with null values being filled for the missing fields in outer joins. Outer joins can be *left, right or full.*

A left outer join means tuples from the left side will be included even when they do not have a match on the right side.

Similarly, a *right* outer join means tuples from the right side will be included even when they do not have a match on the left side.

A full outer join implies tuples from both sides are taken even when they do not have matches.

Q3

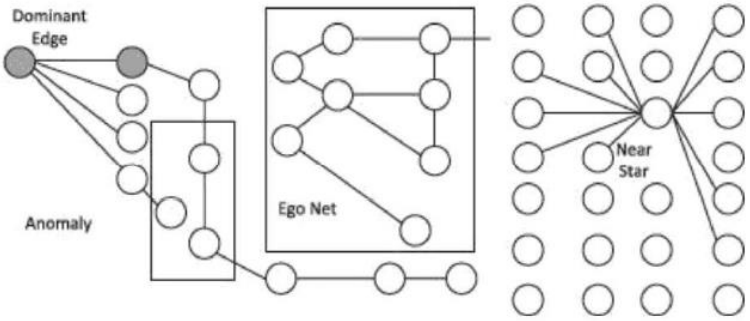
Important metrics are degree (centrality), closeness (centrality), betweenness (centrality) and eigenvector (centrality). Eigenvector consists of elements such as status, rank and other properties. Social graph-network analytics discovers the degree of interactions, closeness, betweenness, ranks, probabilities,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

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

Social network analysis enables detection of an anomaly. An example is detection of one dominant edge which other sub-graphs are follow (succeed). Ego network is another

example. The network structure is such that a given vertex corresponds to a sub-graph where only its adjacent neighbours and their mutual links are included.

The analysis enables spam detection. Spam is discovered by observation of a near star structure. Figure 9.12 shows discovering anomaly, ego-net and spam from the analysis.



Q4.

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

Figure 9.9 shows Pg (v) in-links from Pg (u) and other pages**.**

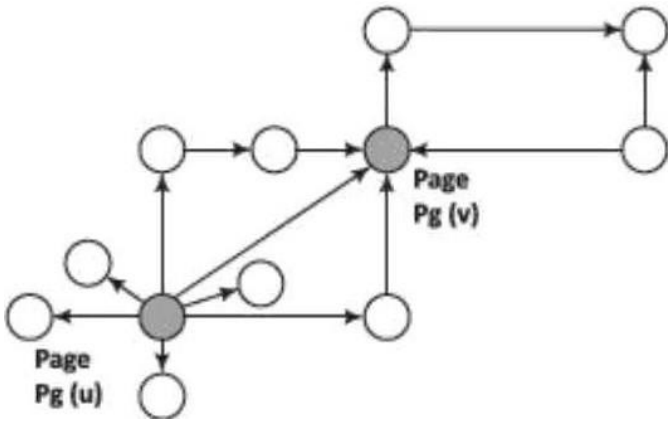


Figure 9.9 Page Pg **(v)** in-links from Pg **(u)** and other pages

Two algorithms to compute page rank are as follows:

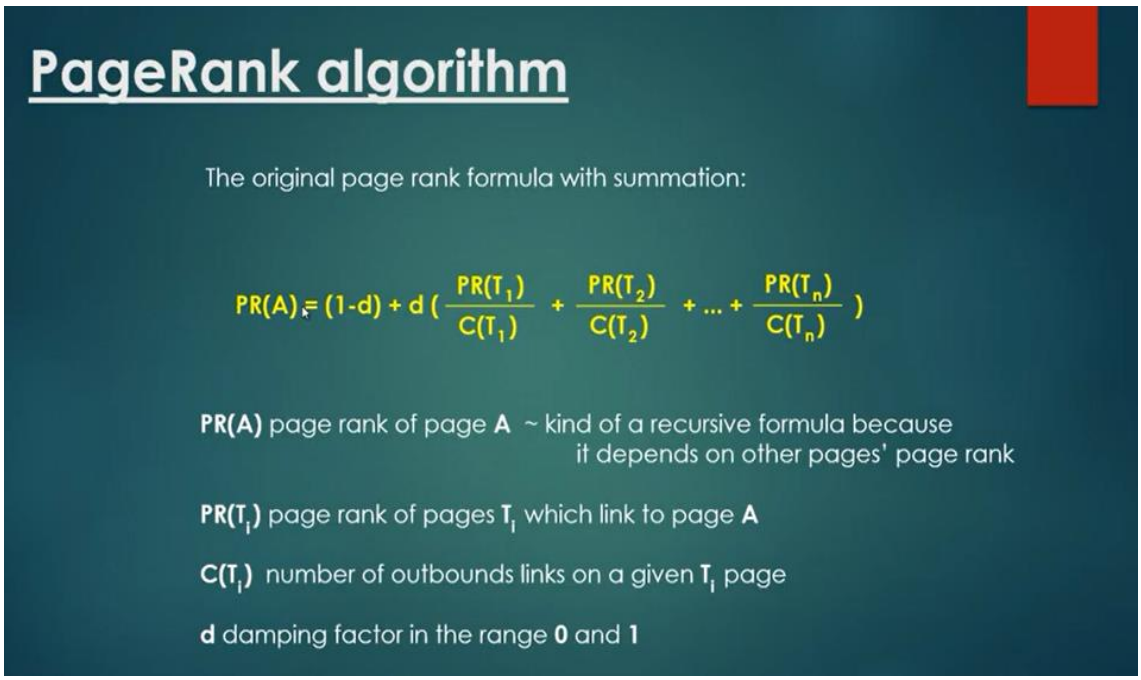
1. PageRank algorithm using the in-degrees as conferring authority

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

$$PR(Pgv) = nc \cdot \sum_{Pgu:Pgu \to Pgv} [PR(Pgu)/N(Pgu)]$$

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

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



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

The algorithm uses the relative authority of the parents (children) and adds a rank for each page from a rank source.

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

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

$$R(v) = \sum_{u \in PA(v)} \left[R(u) / |ch(u)| \right] \tag{9.25}$$

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

Assume that a rank source E exists that is addition to the rank of each page R (v) by a fixed rank value E(v) for Pgv. E(v) is fraction α of [1/|PA(v)|].

An alternative equation is as follows:

$$R(v) = nc \cdot \left\{ (1-\alpha) \sum_{u \in PA(v)} \left[\frac{R(u)}{|ch(u)|} \right] + \alpha \cdot E(v) \right\}. \tag{9.26}$$

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

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

find TF and IDF:

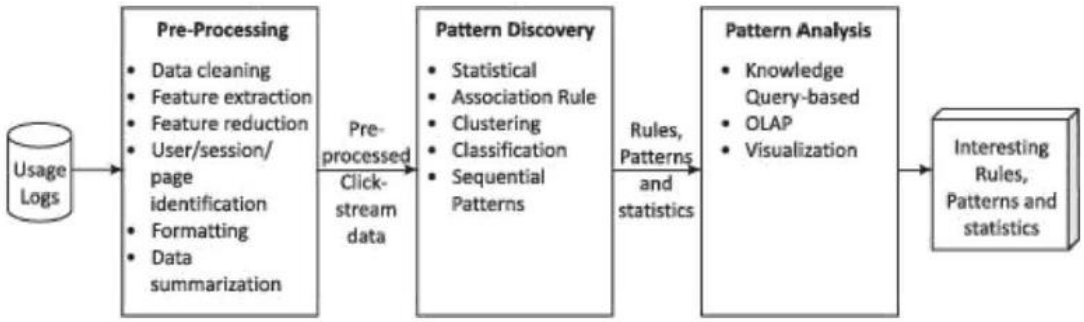
$$TF - IDF(t) = \frac{\text{No. of times t appears in a document}}{\text{Total No. of terms in the document}} \times \log \frac{\text{No. of documents in the collection}}{\text{No. of documents that contain t}}. \tag{9.1}$$

Q5

9.3.1 Web Usage Mining

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

Figure 9.7 shows three phases for web usage mining.



The phases are:

1. Pre-processing - Converts the usage information collected from the various data sources into the data abstractions necessary for pattern discovery.

2. Pattern discovery - Exploits methods and algorithms developed from fields, such as statistics, data mining, ML and pattern recognition.

3. Pattern analysis - Filter outs uninteresting rules or patterns from the set found during the pattern discovery phase.

9.3.3.1 Pre-processing

The common data mining techniques apply on the results of pre-processing using vector space model (Refer Example 9.2). Pre-processing is the data preparation task, which is required to identify:

(i) User through cookies, logins or URL information

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

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

(iv) Page references.

The process deals with (i) extracting of the data, (ii) finding the accuracy of data, (iii) putting the data together from different sources, (iv) transforming the data into the required format and (iv) structure the data as per the input requirements of pattern discovery algorithm.

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

9.3.3.2 Pattern Discovery

The pre-processed data enable the application of knowledge extraction algorithms based on statistics, ML and data mining algorithms.

Statistical techniques They are the most common methods which extract the knowledge about users. They perform different kinds of descriptive statistical analysis (frequency,

mean, median) on variables such as page views, viewing time and length of path for navigational.

Statistical techniques enable discovering:

(i) The most frequently accessed pages

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

(iii) Providing support for marketing decisions

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

Other uses of association rule mining are:

(i) Reveal a correlation between users who visited a page containing similar information.
(ii) Provide recommendations to purchase other products. For example, recommend to user who visited a web page related to a book on data analytics, the books on ML and Big Data analytics also.
(iii) Provide help to web designers to restructure their websites.
(iv) Retrieve the documents in prior in order to reduce the access time when loading a page from a remote site.

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

(i) Establish groups of users showing similar browsing behaviors

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

(iii) Provide personalized web content to users

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

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

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

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

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