CSCI316 – Big Data Mining Techniques and Implementation

REVISION – 2021S1

Big data and Pre-processing Concepts

Why pre-processing can improve the quality of data pipeline?

Big data and Pre-processing Concepts

Why pre-processing can improve the quality of data pipeline?

- Data pre-processing is a required first step before any machine learning machinery can be applied. It consists of data cleaning, data transformation and feature selection.
- Through data cleaning, inconsistent data, missing information, errors, and outliers can be corrected before data are submitted to machine learning model.

Big data and Pre-processing Concepts

Why pre-processing can improve the quality of data pipeline?

- Through data transformation, the training/testing data are scaled and transformed. For example, categorical data can be transformed to numerical data without losing their original characteristics.
- Through feature selection, relevant features from big data can be selected to serve as input to machine learning model.

Big data and Pre-processing Concepts

What is the difference between noise and outliers?

Big data and Pre-processing Concepts

What is the difference between noise and outliers?

- Outliers are observed data that are distant (different) from the remaining observed data. They are also referred to as abnormalities, discordant, deviants and anomalies.
- Noise are corrupted or distorted data containing false information. In other words, noise are mislabelled examples or errors in the value of attributes.

Big data and Pre-processing Concepts

What is the difference between noise and outliers?

- Hence, we can say outliers are data that do not "fit in" with the other data that we are analysing. Outlier can be a valid data point, or it can be noise. As such, outliers often contain interesting and useful information about the underlying system. E.g., in an intrusion detection system, an outlier can be an instance of intrusion.

Big data and Pre-processing Concepts

Why large-scale machine learning is challenging?

Lecture6 – Handling Massive Datasets, slide 38 – 41.

Python programming

Implement from scratch a Python function that takes a list of string values and returns…

Assume that a Python list is defined as follows:

$$X = list(range(10))$$

Using the list comprehension method, implement another list named Y which contains all the even numbers in X. Present the Python code of your implementation. Also present the output of the command “print(Y)”.

Python Programming

Assume that a Python dictionary is defined as follows:

Sort the elements in Word Counts from the highest to lowest counts of the keys. Present the Python code of your implementation.

Decision Tree

Suppose you are designing a DT classifier to processing the following data set:

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Present the process of using InfoGain to split the data set according to the “student” feature. Detail the steps in your computation process.

Decision Tree

Creating a binary decision tree is actually a process of dividing up the input space. A greedy approach is used to divide the space called recursive binary splitting. This is a numerical procedure where all the values are lined up and different split points are tried and tested using a cost function.

The split with the best cost (lowest cost because we minimize cost) is selected. All input variables and all possible split points are evaluated and chosen in a greedy manner based on the cost function.

One possible technique for deciding on splitting of nodes can be done using information gain.

Decision Tree

For example, to design a decision tree classifier of the above-mentioned data set according to the feature 'student' using information gain, we carry out the following processes:

Step 1:

Before the split, we compute the entropy (impurity or uncertainty) of the class 'buysComputer' in order to find the information gain. The formula of entropy is:

$$Entropy = -\sum_{i=1}^{n} p_i \times \log_2 p_i$$

Decision Tree

For example, in the class 'buysComputer' of the data set, there are 7 students and 7 non-students, that is, 9 'yes' and 5 'no'. We split the class into $\text{buysComputer} = \text{'}yes\text{'}$ and $\text{buysComputer} = \text{'}no\text{'}$, and calculate the respective entropy:

The $Entropy(buysComputer = yes) = -\sum_{i=1}^{9} p_i \times \log_2 p_i$

$$= -\frac{9}{14} \times \log_2 \frac{9}{14} = -0.643 \times -0.637 = 0.410$$

The $Entropy(buysComputer = no) = -\sum_{i=1}^{5} p_i \times \log_2 p_i$

$$= -\frac{5}{14} \times \log_2 \frac{5}{14} = -0.357 \times -1.485 = 0.53$$

Decision Tree

The entropy for the class 'buysComputer' is hence,

$$E(buysComputer) = E(buysComputer =' yes') + E(buysComputer =' no')$$
$$= 0.410 + 0.530 = 0.94$$

Decision Tree

Step 2:

Next, we consider the feature ‘student’ and compute the entropy for the respective classes:

Student / buysComputer	yes	no	
Yes	6	3	9
No	1	4	5
	7	7	14

Decision Tree

$$E(buysComputer =' yes', student =' yes') = -\frac{6}{7} \times \log_2 \frac{6}{7}$$

$$= -0.857 \times \log_2 0.857 = 0.191$$

$$E(buysComputer =' no', student =' yes') = -\frac{1}{7} \times \log_2 \frac{1}{7}$$

$$= -0.143 \times \log_2 0.143 = 0.401$$

$$E(buysComputer =' yes', student =' no') = -\frac{3}{7} \times \log_2 \frac{3}{7}$$

$$= -0.429 \times \log_2 0.429 = 0.524$$

$$E(buysComputer =' no', student =' no') = -\frac{4}{7} \times \log_2 \frac{4}{7}$$

$$= -0.571 \times \log_2 0.571 = 0.462$$

Decision Tree

$E(buysComputer, student =' yes') = 0.191 + 0.401 = 0.592$

$E(buysComputer, student =' no') = 0.524 + 0.462 = 0.986$

Step 3:

The information gain for the feature 'student' is hence,

$IG(buysComputer, student) = E(buysComputer) - \sum_{student \in \{yes,no\}}^{n} \frac{|S_{student}|}{|S|} E_{(S_{student})}$

$IG(buysComputer, student) = 0.94 - \left[\frac{7}{14} \times 0.592 + \frac{7}{14} \times 0.986\right]$

$= 0.94 - [0.381 + 0.352] = 0.151$

Naïve Bayes classifier and model evaluation

Explain a method to handle zero counts and a method to handle numerical underflow in the Naïve Bayes classifier. Use example o support your answers.

Lecture5 – BayesClassifier, Classification, Evaluation and Model Enhancement, slide 25, 26.

Naïve Bayes classifier and model evaluation

Why the Naïve Bayes classifier is efficient (e.g., compared with Decision Tree)?

- Bayes can perform quite well, and it doesn't over fit nearly as much so there is no need to prune or process the network.
- Naive bayes does quite well when the training data doesn't contain all possibilities so it can be very good with low amounts of data.

Naïve Bayes classifier and model evaluation

Considering the following confusion matrix, what are the TP, FP, TN, FN, precision, recall and F-measure?

Actual class / Predicted class	buyComputer=yes	buyComputer=no	Total
buyComputer=yes	6954 (TP)	46 (FP)	7000
buyComputer=no	412 (FN)	2588 (TN)	3000
Total	7366	2634	10000

Naïve Bayes classifier and model evaluation

Actual class / Predicted class	C	$\bar{C}$	
C	6954	46	$P(positive\ tuple)$
$\bar{C}$	412	2588	$N(Negative\ tuple)$
Total	P'	N'	All

Precision (Exactness): What % of tuples that the classifier labelled as positive are actually positive.

$$precision = \frac{TP}{TP + FP}$$

Recall (completeness): What % of positive tuples did the classifier labelled as positive? (equal to sensitivity).

$$recall = \frac{TP}{TP + FN}$$

Lecture5 BayesClassificatiion, slide 35 - 39

Naïve Bayes classifier and model evaluation

Actual class / Predicted class	C	$\bar{C}$	
C	6954	46	$P(positive\ tuple)$
$\bar{C}$	412	2588	$N(Negative\ tuple)$
Total	P'	N'	All

F-measure (F1 or F-score): Harmonic mean of precision and recall.

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

Lecture5 BayesClassificatiion, slide 35 - 39

Naïve Bayes classifier and model evaluation

Considering the following confusion matrix, what are the TP, FP, TN, FN, precision, recall and F-measure?

Actual class / Predicted class	buyComputer=yes	buyComputer=no	Total
buyComputer=yes	6954 (TP)	46 (FP)	7000
buyComputer=no	412 (FN)	2588 (TN)	3000
Total	7366	2634	10000

$TP = 6954$ $TN = 2588$

$FP = 46$ $FN = 412$

$$precision = \frac{TP}{TP + FP} = \frac{6954}{6954 + 46} = 0.993$$

$$recall = \frac{TP}{TP + FN} = \frac{6954}{6954 + 412} = 0.944$$

Naïve Bayes classifier and model evaluation

Considering the following confusion matrix, what are the TP, FP, TN, FN, precision, recall and F-measure?

Actual class / Predicted class	buyComputer=yes	buyComputer=no	Total
buyComputer=yes	6954	46	7000
buyComputer=no	412	2588	3000
Total	7366	2634	10000

$$precision = 0.993$$

$$recall = 0.944$$

$$F - Measure = \frac{2 \times precision \times recall}{precision + recall}$$

$$F - Measure = \frac{2 \times 0.993 \times 0.944}{0.993 + 0.944} = 0.968$$

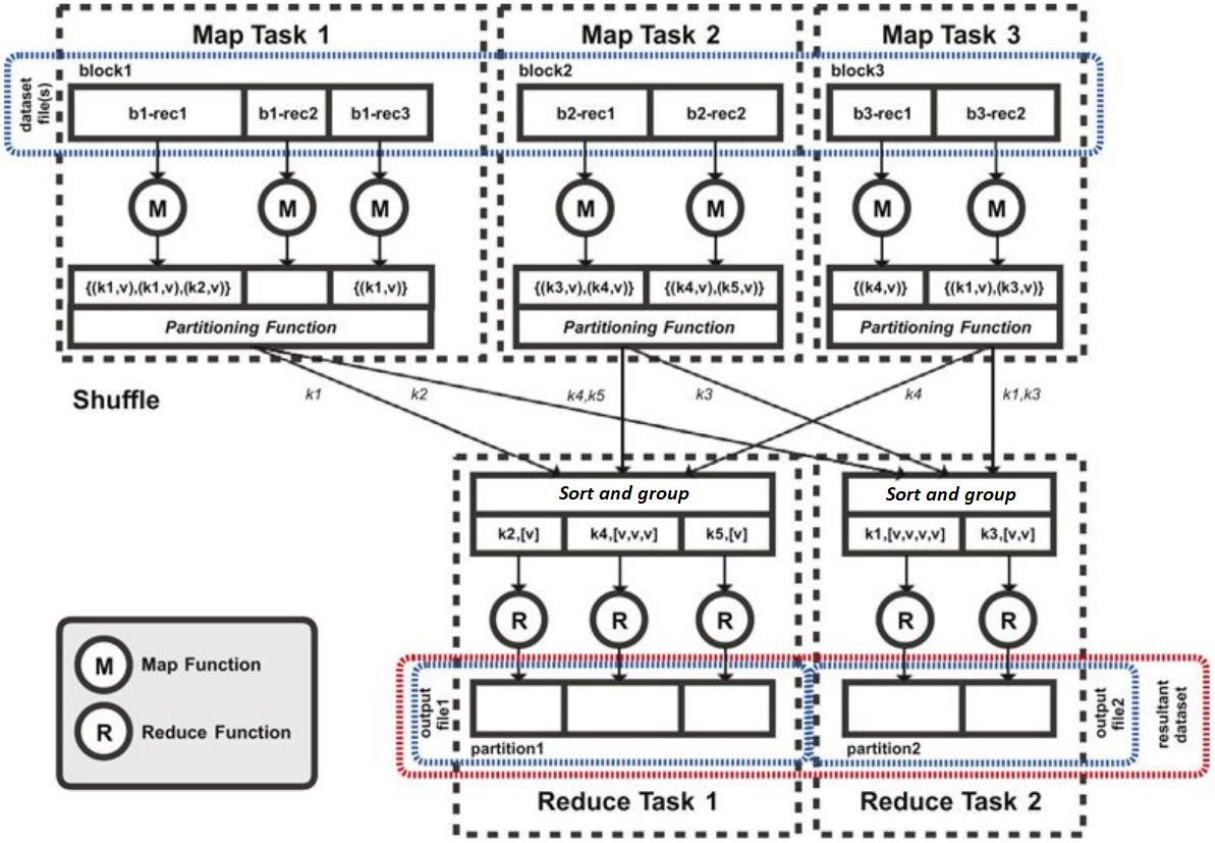
MapReduce and Spark Model

What is a MapReduce?

MapReduce

- Software framework and programming model used for processing a very large amounts of data.
- It works in two phases:
  - Mapping phase
    - In this phase, data is split and mapped into multiple groups.
  - Reduce phase
    - In this phase, the mapped data are shuffle and reduced individual groups.
- MapReduce programs are parallel in nature, hence, they are useful for performing large-scale data analysis using multiple machines in the cluster.

MapReduce and Spark Model



MapReduce and Spark Model

Input Splits:

Data set is divided into fixed-size chunk (block) that is consumed by a single map.

Mapping:

Data in each chunk is passed to a mapping function to produce counts of occurrences of each word, and prepare a list of key-value pair where key is the word, and value is the frequency of occurrences.

Shuffling

Shuffling process will consolidate the relevant records from Mapping phases by clubbing together the same words and accumulate their frequency.

MapReduce and Spark Model

Reducing

In this phase, the output values from the Shuffling phase are aggregated, by combining all the words into a single output, that is, producing a complete dataset.

MapReduce and Spark Model

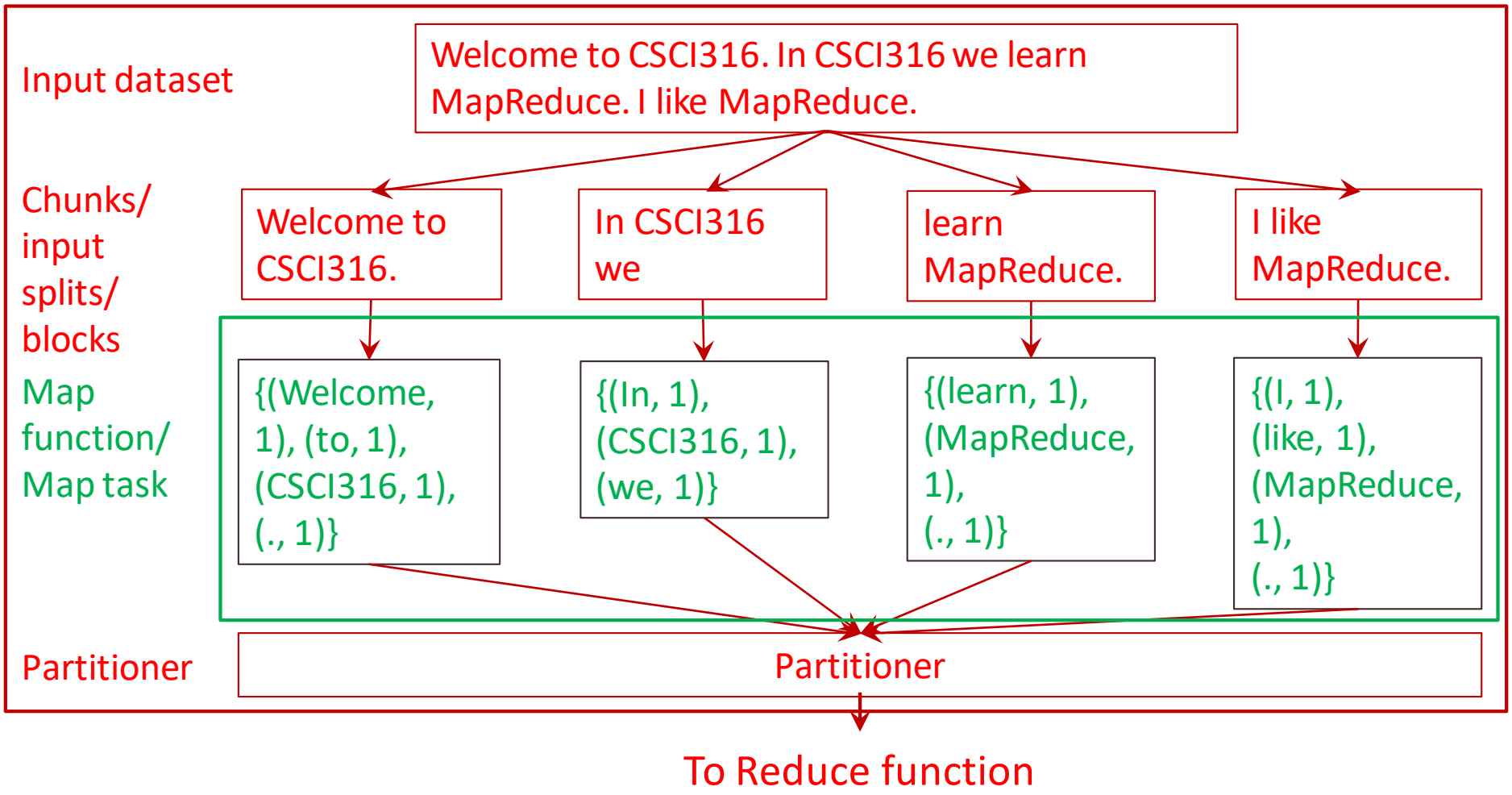
Use an example to explain how the MapReduce model can process a “word count” problem.

MapReduce and Spark Model

The following example depicts a MapReduce function in the Map function.

1. Dataset is split into blocks/chunks of fixed size.
2. Map function is then assigned to process each block, that is, each Map function operates only one block.
3. Each Map function outputs (produces) sets of key-value pair records.
4. The sets of key-value pair records are passed to *Partitioner*, which ensures each key-value pair record is passed to one and only one Reducer.

Map Phase:


Input dataset
Welcome to CSCI316. In CSCI316 we learn MapReduce. I like MapReduce.
Chunks/ input splits/ blocks
Welcome to CSCI316.
In CSCI316 we
learn MapReduce.
I like MapReduce.
Map function/ Map task
{(Welcome, 1), (to, 1), (CSCI316, 1), (., 1)}
{(In, 1), (CSCI316, 1), (we, 1)}
{(learn, 1), (MapReduce, 1), (., 1)}
{(I, 1), (like, 1), (MapReduce, 1), (., 1)}
Partitioner
Partitioner
To Reduce function

MapReduce and Spark Model

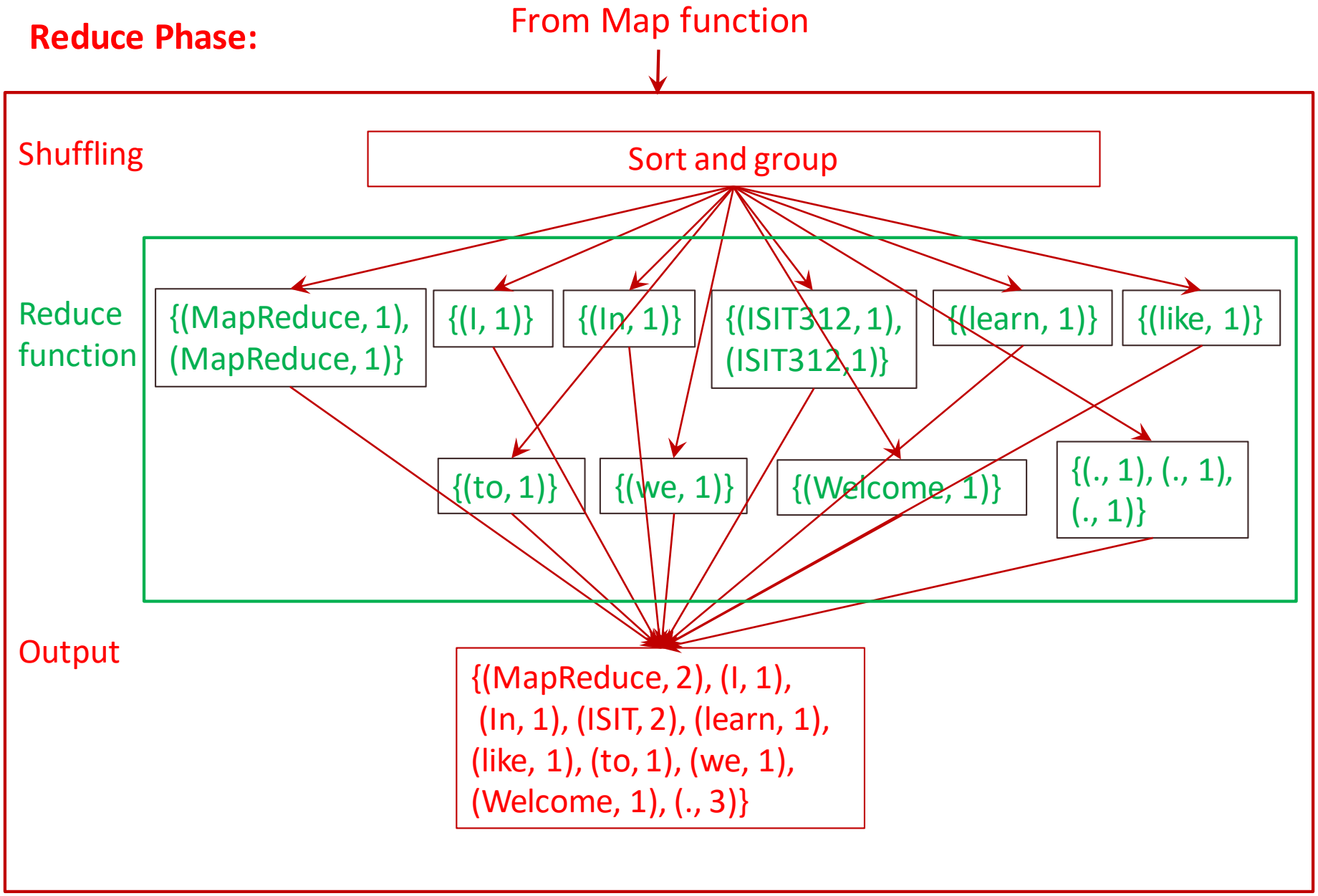
Input to Reduce Phase is the output from Map Phase, that is, the *Partitioner* ensures each key-value pair is passed to one and only one Reducer.

The Reduce will perform a sort and group function to sort and group the key-value pair by the key and accumulate the values for each key.

The Reduce function will then output a set consisting of all the key-value pairs.

Note: A reducer may be receiving input from multiple Map functions.

Reduce Phase:



MapReduce and Spark Model

Explain how the MapReduce model can process the relational-algebra operation “selection”. Use a concrete example to support your answer.

Lecture 6 – Handling Massive Datasets, slide 7-9.

MapReduce and Spark Model

Explain how the MapReduce model can process the relational-algebra operation “selection”. Use a concrete example to support your answer.

For example, the following table (named X) has a column of keys and a column of values:

Key	Value
K1	1
K1	2
K2	3
K2	4

MapReduce and Spark Model

We have the following select statement:

'SELECT key, SUM(value) FROM X GROUPBY key'

1. The Map functions read each row of the table X and produce the following key value pair outputs:
   - (k1, 1), (k1, 2), (k2, 3), (k2, 4)
2. The sets of key-value pair records are passed to *Partitioner*, which ensures each key-value pair record is passed to one and only one Reducer.

MapReduce and Spark Model

3. After the shuffle-and-sort, the input to the Reduce functions are as follows:
   - (k1, [1,2]), (k2, [3,4])
4. The Reduce functions compute the sum for each key, and output the set consisting of all the key-value pairs. Hence, the final output is:
   - (k1, 3), (k2, 7)

MapReduce and Spark Model

Explain how the Spark model extends the MapReduce model.

Please refer to Lecture6 – Handling Massive Datasets, slide 22.

PySpark and Spark MLlib

Assume the following Data Frame is defined in the PySpark shell. Write down your codes in PySpark shell to fulfil the following operations

In [9]:
```
# Create spark session
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('CSCI316-Lab6')\
.config('spark-master', 'local')\
.getOrCreate()

# output the object (spark) information
spark
```

Out[9]:

SparkSession - in-memory

SparkContext

Spark UI

Version

`v3.0.1`

Master

`local[*]`

AppName

`CSCI316-Lab6`

In [10]:
```
# Read a json file from local disk
directory = 'C://Users//japit//Documents//CSCI316'
df_RD = spark.read.format('csv')\
.option('header', 'true')\
.option('inferSchema', 'true')\
.load(directory+'/2010-12-01.csv')

# print the schema
df_RD.printSchema()
```

```
root
 |-- InvoiceNo: string (nullable = true)
 |-- StockCode: string (nullable = true)
 |-- Description: string (nullable = true)
 |-- Quantity: integer (nullable = true)
 |-- InvoiceDate: string (nullable = true)
 |-- UnitPrice: double (nullable = true)
 |-- CustomerID: integer (nullable = true)
 |-- Country: string (nullable = true)
```

PySpark and Spark MLlib

Assume the following Data Frame is defined in the PySpark shell. Write down your codes in PySpark shell to fulfil the following operations

```
In [11]: df_RD.show(2)
```

```
+---------+---------+--------------------+--------+--------------+---------+----------+--------------
+
|InvoiceNo|StockCode|         Description|Quantity|   InvoiceDate|UnitPrice|CustomerID|       Country
|
+---------+---------+--------------------+--------+--------------+---------+----------+--------------
+
|   536365|   85123A|WHITE HANGING HEA...|       6|1/12/2010 8:26|     2.55|     17850|United Kingdom
|
|   536365|    71053| WHITE METAL LANTERN|       6|1/12/2010 8:26|     3.39|     17850|United Kingdom
|
+---------+---------+--------------------+--------+--------------+---------+----------+--------------
+
only showing top 2 rows
```

PySpark and Spark MLlib

Count the unique values in the ‘Invoice’ column.

```
In [12]: df_RD.select('InvoiceNo').distinct().count()
Out[12]: 143
```

PySpark and Spark MLlib

Define a new DataFrame that includes on the ‘StockCode’ and ‘Quantity’ columns of df_RD.

In [30]:
```
from pyspark.sql.functions import struct, col
from pyspark.sql.types import *
# create new dataframe using struct and col
myNewDf = df_RD.withColumn('newStruct', struct('StockCode','Quantity'))
myNewDf = myNewDf.withColumn('StockCode', col('newStruct.StockCode')).\
withColumn('Quantity', col('newStruct.Quantity')).\
drop('InvoiceNo', 'Description', 'InvoiceDate', 'UnitPrice', 'CustomerID', 'Country','newStruct')
myNewDf.show(3)
myNewDf.printSchema()
```

```
+---------+--------+
|StockCode|Quantity|
+---------+--------+
|   85123A|       6|
|    71053|       6|
|   84406B|       8|
+---------+--------+
only showing top 3 rows

root
 |-- StockCode: string (nullable = true)
 |-- Quantity: integer (nullable = true)
```

PySpark and Spark MLlib

Define a new DataFrame that includes on the ‘StockCode’ and ‘Quantity’ columns of df_RD.

In [31]:
```
# create new dataframe using selectExpr
myNewDf2 = df_RD.selectExpr('StockCode as StockCode', 'Quantity as Quantity')
myNewDf2.show(3)
myNewDf2.printSchema()
```

```
+---------+--------+
|StockCode|Quantity|
+---------+--------+
|   85123A|       6|
|    71053|       6|
|   84406B|       8|
+---------+--------+
only showing top 3 rows

root
 |-- StockCode: string (nullable = true)
 |-- Quantity: integer (nullable = true)
```

PySpark and Spark MLlib

Explain the main differences between Spark Data Frame and Pandas Data Frame as data structure.

https://www.analyticsvidhya.com/blog/2016/10/spark-dataframe-and-operations/#:~:text=The%20few%20differences%20between%20Pandas,pandas%20it%20is%20not%20possible.&text=Still%20pandas%20API%20is%20more,to%20perform%20than%20Pyspark%20DataFrame

PySpark and Spark MLlib

Pandas and Spark DataFrame are designed for structural and semistructral data processing. Both share some similar properties. The few differences between Pandas and PySpark DataFrame are:

- Operation on Pyspark DataFrame run parallel on different nodes in cluster but, in case of pandas it is not possible.
- Operations in PySpark DataFrame are lazy in nature but, in case of pandas we get the result as soon as we apply any operation.
- In PySpark DataFrame, we can't change the DataFrame due to it's immutable property, we need to transform it. But in pandas it is not the case.
- Pandas API support more operations than PySpark DataFrame. Still pandas API is more powerful than Spark.
- Complex operations in pandas are easier to perform than Pyspark DataFrame
- In addition to above points, Pandas and Pyspark DataFrame have some basic differences like columns selection, filtering, adding the columns, etc.

Ref: https://www.analyticsvidhya.com/blog/2016/10/spark-dataframe-and-operations/#:~:text=The%20few%20differences%20between%20Pandas,pandas%20it%20is%20not%20possible.&text=Still%20pandas%20API%20is%20more,to%20perform%20than%20Pyspark%20DataFrame

PySpark and Spark MLlib

Spark Mllib is similar to Scikit-Learn in terms of APIs. But what is the most important feature of Spark Mllib?

Lecture6 – Handling Massive Datasets, slide 42 - 58

TensorFlow and ANN

Why the softmax function is suitable for multiple classification but not regression?

- The softmax function is a function that turns a vector of K real values into a vector of K real values that sum to 1. The input values can be positive, negative, zero, or greater than one, but the softmax transforms them into values between 0 and 1, so that they can be interpreted as probabilities. If one of the inputs is small or negative, the softmax turns it into a small probability, and if an input is large, then it turns it into a large probability, but it will always remain between 0 and 1.
- The softmax function can be used in a classifier only when the classes are mutually exclusive (under the assumption that the classes are mutually exclusive).

TensorFlow and ANN

Regression involves processes that estimate the relationships between a dependent variables, also called ‘outcome variable’, and one or more independent variables, also called ‘predictors’. Regression analyse dependencies amongst variables by estimating the effect that changing one independent variable has on the dependent variable while holding all the other independent variables constant.