Sinhgad Institutes

Name of the Student:_____**Roll no:**_____

CLASS:- T.E.[I.T] **Division: A**

Course: - 2019

Subject: 314457: Data Science and Big Data Analytics Laboratory

PART_ B _Assignment No. 03

Marks: _____/10

Date of Performance: ___/____/_____ **Sign with Date: _____**

Part- B

ASSIGNMENT NO: 03

TITLE:

To integrate Python and Hadoop and perform the following operations on forest fire dataset

AIM:

To perform the following operations:

a. Data analysis using the Map Reduce in PyHadoop
b. Data mining in Hive

OBJECTIVE:

1. To understand and apply the Analytical concept of Big data using Python.
2. To apply the Analytical concept of Big data using Python.

Software used: IDLE Shell 3.9.6 (Python 3.9.6)

THEORY:

Text Mining:

Natural languages (English, Hindi, Mandarin etc.) are different from programming languages. The semantic or the meaning of a statement depends on the context, tone and a lot of other factors. Unlike programming languages, natural languages are ambiguous.

Text mining deals with helping computers understand the “meaning” of the text. Some of the common text mining applications include sentiment analysis e.g if a Tweet about a movie says something positive or not, text classification e.g classifying the mails you get as spam or ham etc. R is succinctly described as “a language and environment for statistical computing and graphics,” which makes it worth knowing if you’re dabbling in the data science/art of statistics and exploratory data analysis. R has a wide variety of useful packages.

In R the packages useful in understanding and extracting insights from the text and text mining packages are as follow:

a. RSQLite, 'SQLite' Interface for R
b. tm, framework for text mining applications
c. SnowballC, text stemming library
d. Wordcloud, for making wordcloud visualizations
e. Syuzhet, text sentiment analysis
f. ggplot2, one of the best data visualization libraries
g. quanteda, N-grams

Text preprocessing:

Before we dive into analyzing text, we need to preprocess it. Text data contains white spaces, punctuations, stop words etc. These characters do not convey much information and are hard to process. For example, English stop words like "the", "is" etc. do not tell you much information about the sentiment of the text, entities mentioned in the text, or relationships between those entities. Depending upon the task at hand, we deal with such characters differently. This will help isolate text mining in R on important words.

- Convert the text to lower case, so that words like "write" and "Write" are considered the same word for analysis
- Remove numbers
- Remove English stopwordse.g "the", "is", "of", etc
- Remove punctuation e.g ",", "?", etc
- Eliminate extra white spaces
- Stemming our text

Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form. E.g changing "car", "cars", "car's", "cars'" to "car". This can also help with different verb tenses with the same semantic meaning such as digs, digging, and dig. One very useful library to perform the aforementioned steps and text mining in R is the "tm" package. The main structure for managing documents in tm is called a Corpus, which represents a collection of text documents.

Cleaning text in R:

```
1 # Transform and clean the text
2 library("tm")
3 docs <- Corpus(VectorSource(emailRaw)) # emailRaw is sample file name
```

Transformations are done via the tm_map() function which applies a function to all elements of the corpus. Basically, all transformations work on single text documents and tm_map() just applies them to all documents in a corpus.

A document term matrix is an important representation for text mining in R tasks and an important concept in text analytics. Each row of the matrix is a document vector, with one column for every term in the entire corpus.

Naturally, some documents may not contain a given term, so this matrix is sparse. The value in each cell of the matrix is the term frequency. tm makes it very easy to create the term-document matrix. With the document term matrix made, we can then proceed to build a word cloud for Hillary's emails, highlighting which words are the most frequently made.

WORD CLOUD: A word cloud is a simple yet informative way to understand textual data and to do text analysis.

Word Cloud is another way of representing the frequency of terms in a document. Here, the size of a word indicates its frequency in the document corpus. Load the wordcloud package, as follows:

```
library(wordcloud)
```

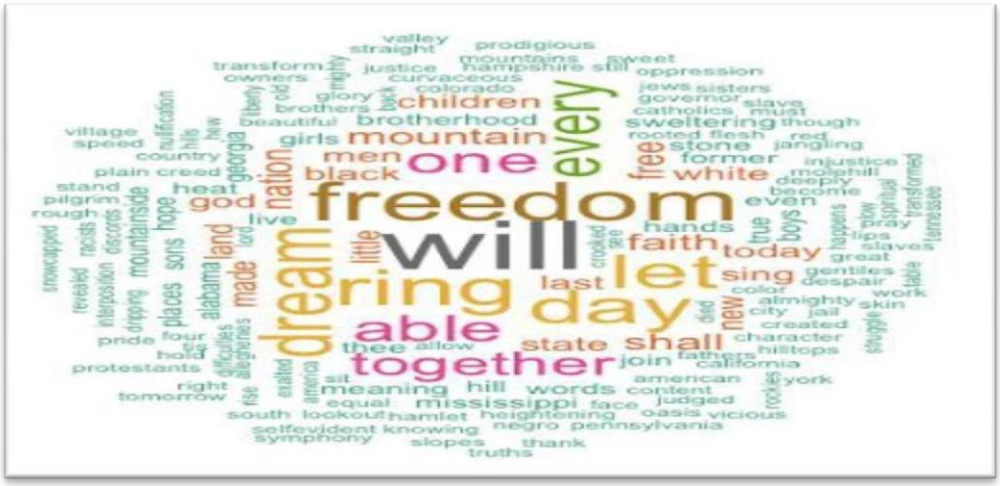
For Word Cloud comprising terms with a frequency greater than 30, use the following command:

wordcloud(names(freq), freq, min.freq=30,colors=brewer.pal(3,"Dark2"))

For a Word Cloud for the 50 words that occur most often, use the command given below:

wordcloud(names(freq), freq, max.words=50,colors=brewer.pal(6,"Dark2"))

Text processing is about extracting useful information from text, which includes basic steps of pre- processing data, stemming the data, representing the corpus using the document term matrix and obtaining the associations between terms. R provides several libraries and functions to efficiently carry out these tasks.



CONCLUSION:

After the study of this assignment we are familiar to integrate Python and Hadoop and perform Data analysis using the Map Reduce in PyHadoop and Data mining in Hive using Python.

Write Short Answers for Following Questions

1. What is the best way to use Hadoop and R together for analysis?
2. Which function gets used for text mining in R?
3. Which package used to create wordcloud in R?

Viava Questions

1. What is supervised and unsupervised learning?
2. How to run map reduce program in R?
3. What are the different techniques used to process text data?

Python Program

https://www.youtube.com/watch?v=O_B7XLfx0ic

Sentiment analysis

```
from textblob import TextBlob
Feedback1 = "The Food at Radison was awesome"
Feedback2 = "The Food at Radison was very good"
blob1 = TextBlob(Feedback1)
blob2 = TextBlob(Feedback2)
print(blob1.sentiment)
print(blob2.sentiment)

>>>
Sentiment(polarity=1.0, subjectivity=1.0)
Sentiment(polarity=0.9099999999999999, subjectivity=0.7800000000000001)
>>>
```

WordCloud

```
# Python program to generate WordCloud
# importing all necessary modules
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import pandas as pd
# Reads 'Youtube04-Eminem.csv' file
df = pd.read_csv(r"Youtube04-Eminem.csv", encoding ="latin-1")
comment_words = ''
stopwords = set(STOPWORDS)
# iterate through the csv file
for val in df.CONTENT:
        # typecaste each val to string
        val = str(val)
        # split the value
        tokens = val.split()

        # Converts each token into lowercase
```

```
        for i in range(len(tokens)):
                tokens[i] = tokens[i].lower()
        comment_words += " ".join(tokens)+" "
wordcloud = WordCloud(width = 800, height = 800,
                                background_color ='white',
                                stopwords = stopwords,
                                min_font_size = 10).generate(comment_words)
# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```