Introduction to Data Science with Apache Zeppelin

Khalid Huseynov
NFLabs, Seoul, South Korea
khalidhnv@nflabs.com

***Abstract* — Apache Zeppelin is a collaborative data analytics and visualization tool for distributed, general-purpose data processing systems such as Apache Spark, Apache Flink, etc. It is a modern web-based tool for the data scientists to collaborate over large-scale data exploration and visualization projects. Its notebook style interpreter enables collaborative analysis sessions sharing between users.**

Keywords — big data; visualization; notebook analytics; open source

I. INTRODUCTION

Data science refers to an interdisciplinary branch of computer science that facilitates extraction of knowledge or insights from large amounts of structured or unstructured data [1, 2]. This process of data discovery can be divided into multiple steps as shown in Figure 1 [3]. Thus, this is an iterative process of data collection, cleaning, analysis, visualization and decision making.

Apache Zeppelin is an open source project for simplifying big data analytics with web-based notebooks that enable interactive data analytics. It has multiple language backends and Apache Spark [4] integration, thus allowing to address various analytic tasks inside notebooks. Moreover it allows interactive visualization and collaboration with ready insights into data. Thus, Apache Zeppelin can cover whole data discovery flow inside a single Zeppelin notebook.

Apache Zeppelin is currently an incubating project under Apache Software Foundation (ASF) [5] with all the following copyright, development process, and community decision making implications. It has more than 70 contributors and its first release of 0.5.0 version happened in July of 2015. Although in an early stage, Apache Zeppelin is growing its user base from industry as well as research communities.

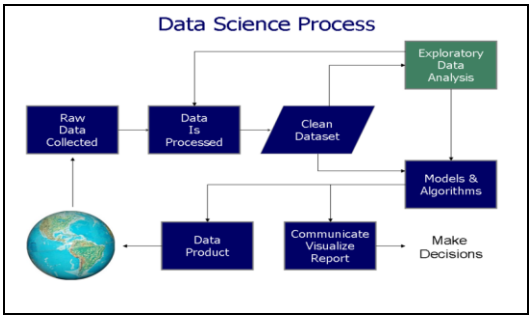

Figure 1 Data Science Process

Supported by NFLabs – Seoul, South Korea (http://www.nflabs.com)

Furthermore, multiple existing solutions address this problem in various ways. For example, Jupyter (IPython) notebook [6] is one of the existing notebook-based solutions that implements python interface with additional computational features known altogether as IPython. Another project known as Spark notebook [7] is forked from Scala notebook project [8] and almost entirely refactored for Massive Dataset Analysis using Apache Spark.

II. ARCHİTECTURE

A. *Multiple Backends*

Apache Zeppelin has pluggable backend architecture in terms of its interpreters. New interpreters can be implemented and plugged in via its *Interpreter* interface. Figure 2 showcases some of the existing interpreters supported by Zeppelin.

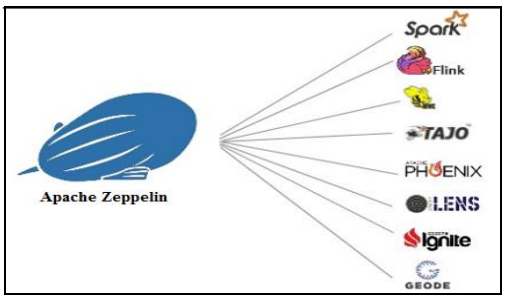

Figure 2 Various backend interpreters

Thus all of the backend frameworks implemented by interpreters can be used inside Zeppelin. Moreover, one Zeppelin notebook may contain different paragraphs each using its own language/interpreter.

B. *Display System*

Since Zeppelin notebooks are web-based, Zeppelin has its *webapp* front-end and back-end server. This architecture is shown in Figure 3.

Zeppelin *webapp* client communicates with back-end server using HTTP REST and *websocket* APIs. Further, server communicates with the interpreter processes. For example, Spark interpreter is just one of the interpreter processes running. According to clients query, server calls corresponding interpreter with a query to interpret the code. Note that interpreter initialization is lazy and isn't started unless client submitted query for it.

Currently, Zeppelin *webapp* front-end supports visualization by the following four different means: pure text, html, tables, and *angular* objects. Angular objects allow Zeppelin to build interactive visualizations inside the notebook.

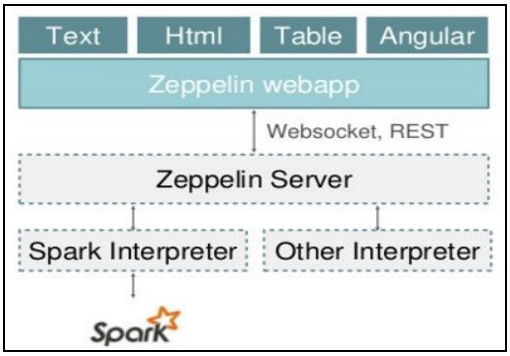

Figure 3 Apache Zeppelin Architecture

III. ZEPPELİN DEMO ANALYTİCS

In this section we will get into real analytics showcase using Apache Zeppelin. After installation [9], Zeppelin includes default showcase notebook with analytics on banking dataset [10].

A. *Prepare Data*

In Zeppelin we can download data using simple *shell* script as if typing in *terminal.* This is realized by the means of *shell* interpreter. Thus, the following script in Figure 4 can download our dataset.

```
%sh
rm ~/bank.zip
rm -rf  ~/data
cd ~
wget http://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank.zip
mkdir data
unzip bank.zip -d data
rm bank.zip

hadoop fs -put  ~/data/bank-full.csv .
hadoop fs -ls -h bank-full.csv
```

Figure 4 Prepare data

The first line with *%sh* implies that shell interpreter is going to be used. Lines 2~3 clean up folder in case previous data exists there. Then we can download data using *wget* command followed by unzipping and putting data into HDFS [11]. Now our data is ready for further processing.

B. *Loading Data into Spark*

Once our dataset is in HDFS we can access it and load into Spark specific RDD format [12]. Figure 5 represents the script written in default Scala Spark interpreter in Zeppelin.

The script starts with creation of Spark context. Further, the input file is mentioned and Bank class representing an object from dataset is created. Then according to structure of Bank class we can load whole dataset using similar template. As you can see, every element of the dataset has five fields: age, job, marital status, education, balance. Finally, this table is registered under "bank" name.

```
//import sys.process._
// sc is an existing SparkContext.
val sqlContext = new org.apache.spark.sql.SQLContext(sc)


val bankText = sc.textFile("bank-full.csv")

case class Bank(age: Integer, job: String, marital: String, education: String, balance: Integer)

val bank = bankText.map(s => s.split(";")).filter(s => s(0) != "\"age\"").map(
    s => Bank(s(0).toInt,
            s(1).replaceAll("\"", ""),
            s(2).replaceAll("\"", ""),
            s(3).replaceAll("\"", ""),
            s(5).replaceAll("\"", "").toInt
        )
)
// toDF() works only in spark 1.3.0.
// For spark 1.1.x and spark 1.2.x,
// use below instead:
// bank.registerTempTable("bank"
bank.toDF().registerTempTable("bank")
```

Figure 5 Loading Data into Spark

C. *Quering Data*

Once we loaded dataset into structured table, we can do various kinds of queries using SQL interpreter of Zeppelin. Suppose we want to get statistics about bank users who are under 30. SQL query from Figure 6 can be used in that case.

```
%sql
select age, count(1) value
from bank
where age < 30
group by age
order by age
```

Figure 6 Query 1 - distribution under 30

First line with *%sql* refers to the usage of SQL interpreter. The results of this query are shown in Figure 7.

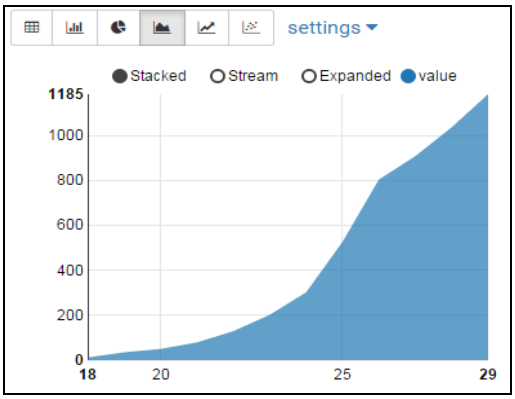

Figure 7 Results of Query 1.

This is *graph* view of the results. Zeppelin also provides *table*, *histogram*, *pie chart* as well as other views. This view is pluggable, thus new types of visualizations can be added by the users.

Next we can query users by *age* and *marital* status as shown in Figure 8.

```
%sql
select age, count(1) value
from bank
where marital="${marital=single,single|divorced|married
}"
group by age
order by age
```

marital

single

Figure 8 Query 2 - by age and marital status

As you can see, the field for marital status is obtained from dropdown menu box. The results of this query are shown in Figure 9.

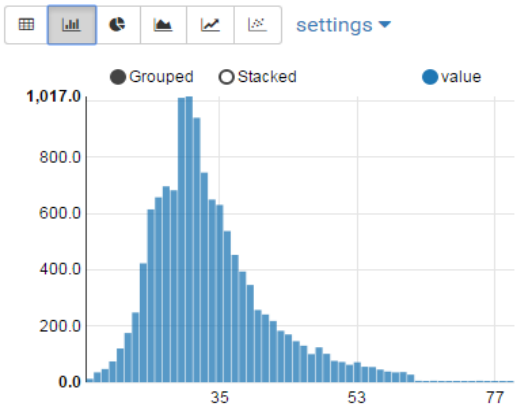

Figure 9 Results of Query 2

These results are shown in *histogram* view. Note that the view can interactively change using *settings* menu. For more insights into Zeppelin workflow you can install it from [9] or visit sharing platform for it discussed in next section.

IV. ZEPPELİNHUB - ZEPPELİN COLLABORATİON PLATFORM

Once you have Zeppelin notebooks, you may need to share or collaborate on them with other parties. For this purpose, ZeppelinHub [13] platform is created. It’s currently in private beta testing state and accepting user applications for beta stage. The collaboration on analytic results is specifically useful inside large companies for decision making in marketing or future company development strategies.

CONCLUSİON AND FUTURE WORK

After going through the data discovery workflow with Apache Zeppelin, we saw Zeppelin addressing all the stages of data discovery process inside a single notebook. Apache Zeppelin supports various back-end interpreters with various front-end visualizations. Moreover, ZeppelinHub lets Zeppelin notebooks to be shared and collaborated on.

REFERENCES

[1] V. Dhar. "Data science and prediction". Communications of the ACM 56 (12): 64. 2013. Vol.56, №12, pp.64, 2013

[2] Jeff Leek. "The key word in "Data Science" is not Data, it is Science". Simply Statistics. 2013

[3] Wiki reference: https://en.wikipedia.org/wiki/Data_science

[4] Apache Spark: http://spark.apache.org/

[5] Apache Software Foundation: http://www.apache.org/

[6] Jupyter (IPython) notebook: http://ipython.org/notebook.html

[7] Spark notebook: https://github.com/andypetrella/spark-notebook

[8] Scala notebook: https://github.com/Bridgewater/scala-notebook

[9] Apache Zeppelin: https://zeppelin.incubator.apache.org/download.html

[10] S. Moro, R. Laureano and P. Cortez. Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology. In P. Novais et al. (Eds.), Proceedings of the European Simulation and Modelling Conference - ESM'2011, pp. 117-121, Guimarães, Portugal, October, 2011. EUROSIS.

[11] Shvachko, et al. "The hadoop distributed file system." Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on. IEEE, 2010.

[12] Zaharia, et al. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing." Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association, 2012.

[13] ZeppelinHub – Zeppelin notebooks collaboration platform: https://www.zeppelinhub.com/