

User Manual  
Big Data Interoperable Framework (BDIF)  
Version 1.0

## Table of Contents

1.	Introduction .....	3
2.	Synaptic Package Manager .....	3
3.	Java Installation.....	4
4.	Scala Installation .....	5
5.	SSH Installation and Configuration .....	6
6.	Hadoop Installation.....	7
7.	BDIF Version 1.0 Installation.....	7
	BDIF Version 1.0 Schema Summary and Installation .....	7
	Compiling BDIF Schema .....	9
8.	Mahout Installation.....	10
9.	OpenCV & JavaCV installation.....	10
10.	LIRE Installation.....	11
11.	BDAS Spark – Scala IDE Setup .....	11
12.	Eclipse IDE Setup .....	11
13.	Validating BDIF With Use Cases .....	13
13.1.	Extracting Visual Image Features using LIRE With BDIF Version 1.0 Interoperability Tool .	13
13.2.	Image Concept Classification Using BDIF Version 1.0.....	16
13.3.	Image Concept Classification Using BDAS Spark.....	20

## 1. Introduction

We have designed and developed a first-of-a-kind robust open-source solution for interoperability between Big Data analytics tools that promises significant impact by offering an extensible framework that could be re-used and built upon by a wider community of stakeholders. We implemented our big data analytics interoperability solution, called the Big Data Interoperability Framework (BDIF), by building upon and extending the Apache Avro foundation tools.

BDIF, written as an extension of Avro, relies on schemas. When BDIF data is read, the schema used when writing it is always present and accessible. This permits each datum to be written with no per-value overheads, making serialization both fast and small. This also facilitates use with dynamic, scripting languages, since data, together with its schema, is self-descriptive. In BDIF, we have designed and created support for the following advanced data formats: vectors, arrays, images, records, graphs, multidimensional arrays, multidimensional images, and spatio-temporal images, or image sequences. Each advanced data format can be represented by the family of primitive and complex data types listed above. This universal collection of advanced data formats, in our assessment, supports a variety of data representations used across the spectrum of data analytics tools surveyed in our research

In this user manual we detail the installation and configuration of BDIF and the support tools required to validate interoperability between a few big data analytic tools such as LIRE, Mahout, and BDAS-SPARK. We discuss installation and configuration of the required tools / software packages in-order to validate our proposed design implementation. A summary of the environments, OS, and tools used in our big data interoperable framework implementation and validation is shown in Table 1-1.

**Table 1-1: Environments and Tools Used for BDIF validation**

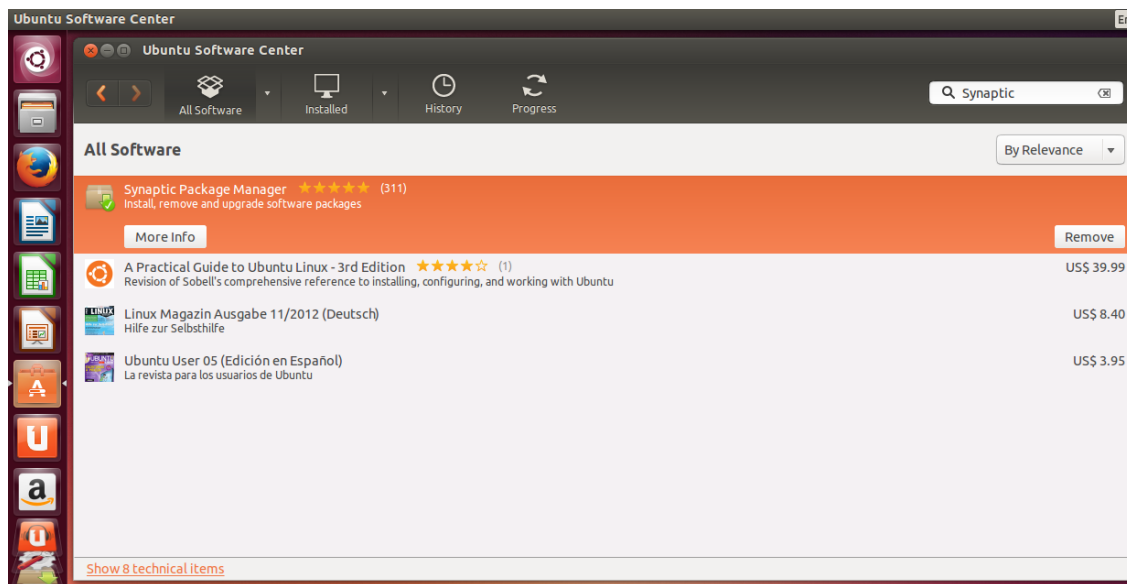
Operation System	:	Ubuntu 12.04
Programming Languages	:	Java, Scala
Frameworks	:	LIRE, JavaCV, OpenCV
Others	:	Hadoop, Mahout, Spark,
Integrated Development Environment [IDE]	:	Scala IDE, Eclipse IDE

The installation and configuration of the tools employed in the BDIF use cases is discussed in the following sections.

## 2. Synaptic Package Manager

**Synaptic Package Manager** is a graphical front-end to apt, the package management system in Ubuntu. It combines the point-and-click simplicity of the graphical user interface with the power of the *apt-get* command line tool. The user can install, remove, configure, or upgrade software packages, browse, sort and search the list of available software packages, manage repositories, or upgrade the whole system. The user can queue up a number of actions before executing them. Synaptic will inform the user about dependencies (additional packages required by the software package you have chosen) as well as conflicts with other packages that are already installed on your system.

The user can download and install Synaptic package manager from the Ubuntu Software center.



### 3. Java Installation

The next steps involves install of the Java environment on an existing OS (Ubuntu 12.04 in our case). It is important to choose the right version of java that is compatible with all the tools / software packages that are considered during this implementation. Oracle java 6 is a stable and compatible version with our proposed solution since the tools were tested and proven with this version of java.

Due to license issues, Ubuntu will no longer distribute Oracle's JDK and JRE. Also previous versions supplied on PPAs suffer from security issues and are therefore are not recommended for installation on a Ubuntu system. There are no more supported java releases from Ubuntu. Ubuntu officially supports OpenJDK and OpenJRE implementation of Java which is the base for Oracle's own implementation. In order to install Oracle java 6 the user needs to add new PPAs to the Ubuntu repository. The following commands should be executed to install Oracle java 6 on Ubuntu.

```
sudo add-apt-repository ppa:webupd8team/java
```

```
hadoop@hadoop-VirtualBox: ~  
hadoop@hadoop-VirtualBox:~$ sudo add-apt-repository ppa:webupd8team/java
```

*sudo apt-get update*

```
hadoop@hadoop-VirtualBox: ~  
hadoop@hadoop-VirtualBox:~$ sudo apt-get update
```

*sudo apt-get install oracle-java6-installer*

```
hadoop@hadoop-VirtualBox: ~  
hadoop@hadoop-VirtualBox:~$ sudo apt-get install oracle-java6-installer
```

After successful installation of java, the version of java needs to be verified. The user should be able to view version of Java as shown in the following screenshot.

*Java -version*

```
hadoop@hadoop-VirtualBox: ~  
hadoop@hadoop-VirtualBox:~$ java -version  
java version "1.6.0_45"  
Java(TM) SE Runtime Environment (build 1.6.0_45-b06)  
Java HotSpot(TM) Client VM (build 20.45-b01, mixed mode)  
hadoop@hadoop-VirtualBox:~$
```

## 4. Scala Installation

Scala is an object-functional programming and scripting language for general software applications, statically typed, designed to concisely express solutions in an elegant, type-safe and lightweight manner. This programming language is required for the Spark Programming environment. The user can install core Scala using following command:

*sudo apt-get install scala*

```
hadoop@hadoop-VirtualBox: ~  
hadoop@hadoop-VirtualBox:~$ sudo apt-get install scala
```

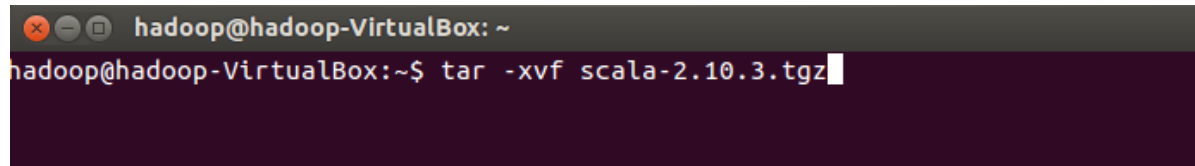
If the user desires to download the source code of scala, the source code can be downloaded from <http://www.scala-lang.org/files/archive/scala-2.10.3.tgz>. followed by extraction of the tar ball.

`wget http://www.scala-lang.org/files/archive/scala-2.10.3.tgz`



```
hadoop@hadoop-VirtualBox: ~  
hadoop@hadoop-VirtualBox:~$ wget http://www.scala-lang.org/files/archive/  
scala-2.10.3.tgz
```

`tar -xvf scala-2.10.3.tgz`



```
hadoop@hadoop-VirtualBox: ~  
hadoop@hadoop-VirtualBox:~$ tar -xvf scala-2.10.3.tgz
```

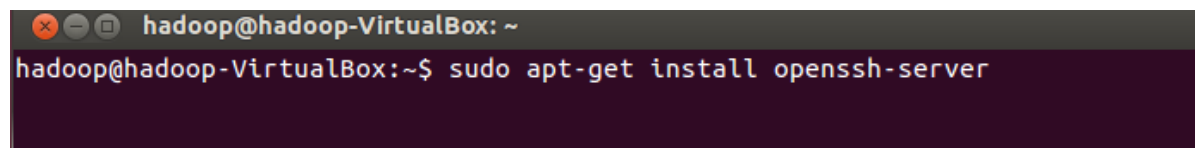
## 5. SSH Installation and Configuration

OpenSSH is a FREE version of the SSH connectivity tools that offers more secure options to telnet, rlogin, and ftp. OpenSSH encrypts all traffic (including passwords) to effectively eliminate eavesdropping, connection hijacking, and other attacks. Additionally, OpenSSH provides secure tunneling capabilities and several authentication methods, and supports all SSH protocol versions.

The OpenSSH suite replaces rlogin and telnet with the ssh program, rcp with scp, and ftp with sftp. Also included is sshd (the server side of the package), and the other utilities like ssh-add, ssh-agent, ssh-keysign, ssh-keyscan, ssh-keygen and sftp-server.

SSH is a pre-requisite for hadoop installation. By default OpenSSH is available with most of the linux boxes. If OpenSSH is not installed, the user should install it as follows:

`sudo apt-get install openssh-server`



```
hadoop@hadoop-VirtualBox: ~  
hadoop@hadoop-VirtualBox:~$ sudo apt-get install openssh-server
```

`ssh localhost` ( *close the connection after successful ssh by typing the exit command* )

```
hadoop@hadoop-VirtualBox: ~  
hadoop@hadoop-VirtualBox:~$ ssh localhost  
Welcome to Ubuntu 13.10 (GNU/Linux 3.11.0-12-generic i686)  
  
* Documentation:  https://help.ubuntu.com/  
  
296 packages can be updated.  
114 updates are security updates.  
  
Last login: Mon Mar 17 17:21:12 2014 from hadoop-virtualbox  
hadoop@hadoop-VirtualBox:~$ exit  
logout  
Connection to localhost closed.  
hadoop@hadoop-VirtualBox:~$
```

## 6. Hadoop Installation

This section describes how to set up and configure a **single-node / stand-alone** Hadoop installation so that the user can quickly perform simple operations in distributed fashion using Hadoop MapReduce and the Hadoop Distributed File System (HDFS). Hadoop installation requires some pre-requisites such as Java, SSH.

For quick reference refer [http://hadoop.apache.org/docs/r1.2.1/single\\_node\\_setup.html](http://hadoop.apache.org/docs/r1.2.1/single_node_setup.html)

## 7. BDIF Version 1.0 Installation

### BDIF Version 1.0 Schema Summary and Installation

Our current BDIF schema design supports pairwise interoperability of analytic tools, such that the user is provided the core functionalities to transfer intermediate data between pairs of analytics tools and can easily extend the schema design to new existing or future identified analytic tools. The detailed schema diagram of the BDIF framework is illustrated in the Figures 7-1, and 7-2 below. Figure 7-1 shows the pairwise interoperability between analytic tools enabled in our current BDIF Version 1.0 with the flow of processes used in BDIF to achieve pairwise interoperability between two big data analytics tools.

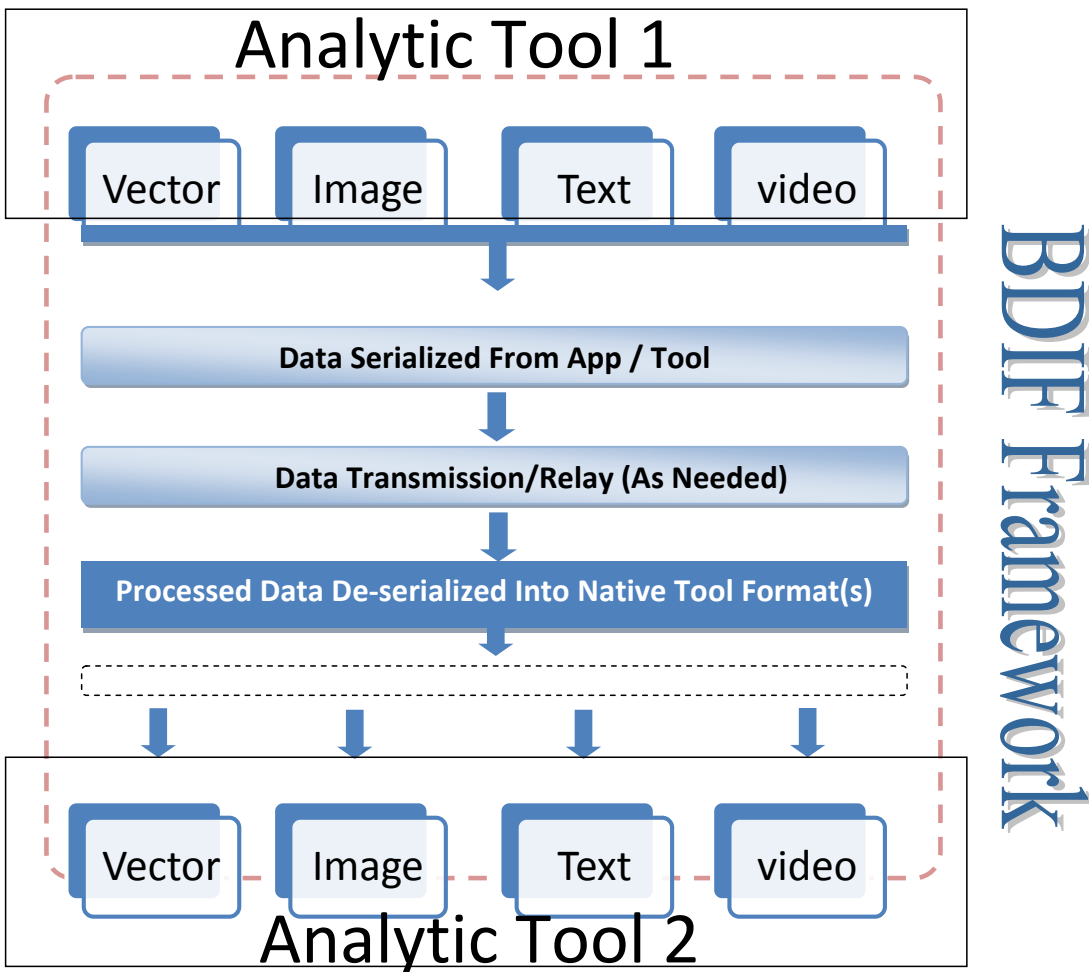


Figure 7-1 Illustration of the steps in our BDIF Version 1.0 that allows interoperability between two big data analytics tools

Big Data Interoperability Framework (BDIF) is built upon and an extension of the Apache Avro foundation tools. AVRO is an open source project under Apache license. Since Avro is implemented in the java programming language, the Avro tools are available in terms of jar files.

A screenshot of the BDIF schema implementation in Java illustrating interoperability between analytic tool pairs such as Mahout and LIRE, JavaCV/OpenCV and BDAS/SPARK, and other tools is shown in Figure 7-2 below.



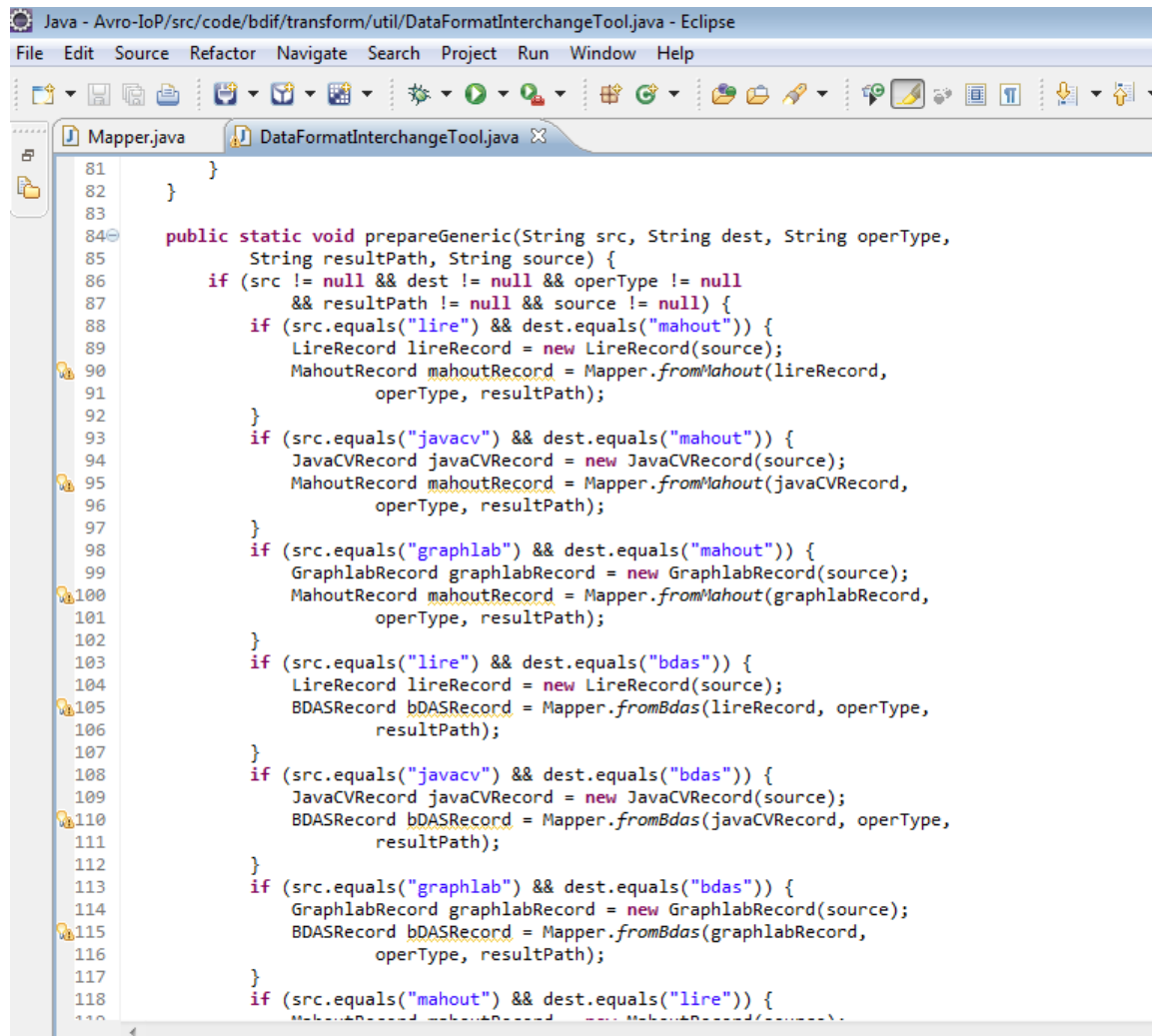


Figure 7-2 Screen capture of the BDIF Version 1.0 Schema Model implementation

The user can download Avro from any one of the Apache mirrors: including the one below:

<http://mirrors.ibiblio.org/apache/avro/stable/java/>

The following 3 jar files are sufficient for the BDIF implementation

<http://mirrors.ibiblio.org/apache/avro/stable/java/avro-compiler-1.7.6.jar>

<http://mirrors.ibiblio.org/apache/avro/stable/java/avro-1.7.6.jar>

<http://mirrors.ibiblio.org/apache/avro/stable/java/avro-tools-1.7.6.jar>

### Compiling BDIF Schema

To compile the BDIF schema and generating java classes the user should execute following command in a terminal window:

`java -jar avro-tools-1.7.5.jar compile schema FeatureSchema.avsc src`

This will generate the java class called "Features.java" into a folder **src** with package name "code.bdif.data.avro".

## 8. Mahout Installation

Apache Mahout is an open source project for scalable Machine Learning. To install and configure this project , the user should have Hadoop on the machine as a pre-requisite.

Download Mahout from <http://archive.apache.org/dist/mahout/0.9/mahout-distribution-0.9.tar.gz>

And extract the tar ball to accessible folder (in which the user should have write permissions).

## 9. OpenCV & JavaCV installation

Installation of OpenCV and JavaCV involves a lot of dependencies installations. To make it easy we have created a script file to install the entire JavaCV and OpenCV environments. The Script file is shown below :

```
arch=$(uname -m)
if [ "$arch" == "i686" -o "$arch" == "i386" -o "$arch" == "i486" -o "$arch" == "i586" ]; then
flag=1
else
flag=0
fi
echo "Installing OpenCV 2.4.3"
mkdir OpenCV
cd OpenCV
echo "Removing any pre-installed ffmpeg and x264"
sudo apt-get -y remove ffmpeg x264 libx264-dev
echo "Installing Dependences"
sudo apt-get update
sudo apt-get -y install libopencv-dev
sudo apt-get -y install build-essential checkinstall cmake pkg-config
sudo apt-get -y install libtiff4-dev libjpeg62-dev libjasper-dev
sudo apt-get -y install libavcodec-dev libavformat-dev libswscale-dev libdc1394-22-dev libxine-dev
libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libv4l-dev
sudo apt-get -y install libtbb-dev
sudo apt-get -y install libfaac-dev libmp3lame-dev libopencore-amrnb-dev libopencore-amrwb-dev
libtheora-dev libvorbis-dev libxvidcore-dev
sudo apt-get -y install libav-tools
echo "Downloading x264"
```

## 10. LIRE Installation

The LIRE (Lucene Image REtrieval) library provides a simple way to retrieve images and photos based on their color and texture characteristics. LIRE creates a Lucene index of image features for content based image retrieval (CBIR). Several different low level features are available, such as MPEG-7 *ScalableColor*, *ColorLayout*, and *EdgeHistogram*, *Auto Color Correlogram*, *PHOG*, *CEDD*, *JCD*, *FCTH*, and many more. Furthermore simple and extended methods for searching the index and result browsing are provided by LIRE. LIRE scales well up to millions of images with hash based approximate indexing. The LIRE library and the LIRE Demo application as well as the entire source are available under the Gnu GPL license.

The user can download the latest version of LIRE from <https://lire.googlecode.com/files/Lire-0.9.3.zip> , extract the zip file and create a java project from this extracted source. During LIRE functional testing, we identified some bugs and fixed those issues. After these bug fixes we exported the revised LIRE source into a binary jar file. The same jar file is made available to our BDIF proposed solution as part of library.

## 11. BDAS Spark – Scala IDE Setup

Apache Spark is a fast and general-purpose cluster computing system. It provides high-level APIs in Scala, Java, and Python that make parallel jobs easy to write, and an optimized engine that supports general computation graphs. It also supports a rich set of higher-level tools including Shark (Hive on Spark), MLlib for machine learning, GraphX for graph processing, and Spark Streaming. The source code & sample programs can be downloaded from the Apache Spark download page.

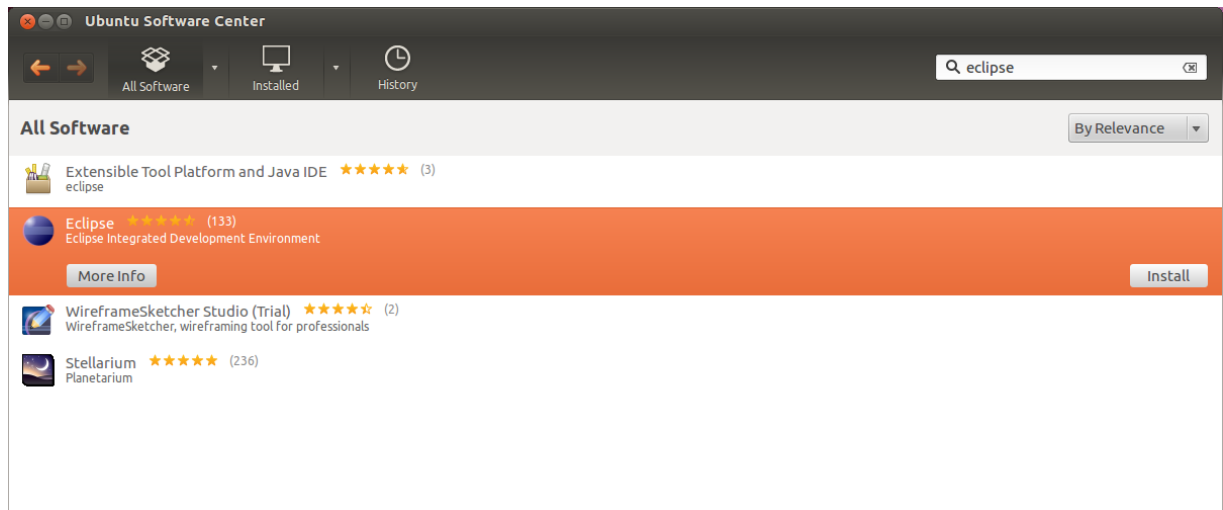
To test the functionality of Spark, we used a compiled version of Spark. The jar file is downloaded from <http://spark-java.googlecode.com/files/spark-0.9.9.4-SNAPSHOT.zip>

## 12. Eclipse IDE Setup

Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Written mostly in Java, Eclipse can be used to develop applications.

To install Eclipse:

1. Open Ubuntu Software Center
2. Type word “eclipse” in the search box
3. Select eclipse from the search result and click on install button
4. To open eclipse type “eclipse” in the Terminal from any directory



## 13. Validating BDIF With Use Cases

To validate the feasibility of our BDIF solution, we have chosen a complex classification task as our use-case. Our use-case involves visual media indexing and concept understanding of unstructured image data, for the purpose of visual media information retrieval. We have used the ImageCLEF dataset as a benchmark for our use case. ImageCLEF was launched in 2003 as part of the Cross Language Evaluation Forum (CLEF) with the goal to provide support for the evaluation of methods used to generate the automatic annotation of images with concepts. We selected the ImageCLEF Visual Concept Detection and Annotation Task (VCDT) as our use-case. Our objective was to show the utility of BDIF in being able to seamlessly and with minimal user-effort integrate two or more big data analytics tools to accomplish visual image concept indexing, classification and retrieval, and demonstrate this on the ImageCLEF VCDT dataset. Note that our focus is on promoting interoperability of analytics tools using BDIF, rather than develop new algorithms in the BDIF framework; therefore we use and integrate existing algorithms in the analytics toolboxes to illustrate the power of BDIF in developing a multimedia visual concept classification solution.

We use a combination of global and local image descriptors for robust visual image indexing and classification. Image-based analytic tools such as JavaCV, OpenCV, or LIRE contain a rich collection of functions for image processing, and extraction of image-derived features. In addition, a robust classifier would be needed to classify an image using the automatically derived image descriptors. Big data analytic tools such as Mahout or BDAS contain a collection of classifier algorithms. We use BDIF to interoperate a visual analytic tool (LIRE) with a classification analytic tool (Mahout or BDAS), to accomplish end to end visual image indexing and classification, and demonstrate this on the ImageCLEF VCDT dataset.

For our visual concept detection use-case, we downloaded a sub-set of images (320 images) from the ImageCLEF VCDT task from [ftp://ftp.fraunhofer.de/institute/idmt/ImageCLEF\\_collections.zip](ftp://ftp.fraunhofer.de/institute/idmt/ImageCLEF_collections.zip), and comprised of 5 concept classes of the 99 concept classes in the dataset. Our goal is to demonstrate the efficacy of BDIF by integrating the visual feature extraction tools from LIRE with the classification capabilities of Mahout or BDAS-SPARK, and illustrate how visual image-based concept classification can be achieved with BDIF without having to write new algorithms in a new tool.

### 13.1. Extracting Visual Image Features using LIRE With BDIF Version 1.0 Interoperability Tool

The user should execute the ExtractBoVW functional class from Eclipse IDE to generate a bag of visual words features descriptor of an image. If the user wants to extract only bag of visual words, then the createAvroRecord(null) method from main method should be used. If additional features from an image are desired, such as Simple Color Histogram, and Edge Histogram along with the bag of visual words along then the function calls should be as follows:

createAvroRecord(CEDD) -- To generate BoVW and CEDD features

createAvroRecord(SCH) -- To generate BoVW and Edge Histogram features

createAvroRecord(EH) -- To generate BoVW and Simple Color Histogram features

**Note:** The user should ensure that the correct path for the image data directory should be specified ( the full of path of dataset should end with "/" )

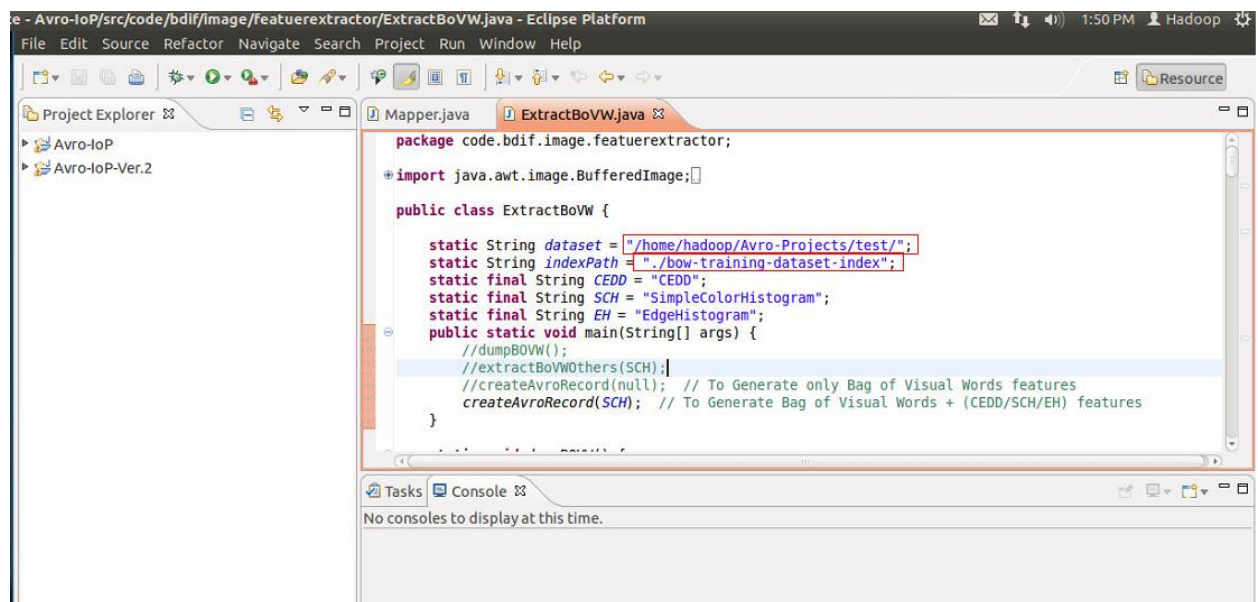


Figure 13-1 : Dataset path & index path variables

The user should make sure the index path is created in the project root directory. The user should select the project root and select F5 or right click on the project root and select refresh in order to view the index path (folder).

The user should locate and select the file: ***"features.avro"*** in project root directory. This file is the input file for the BDIF Interoperability framework.

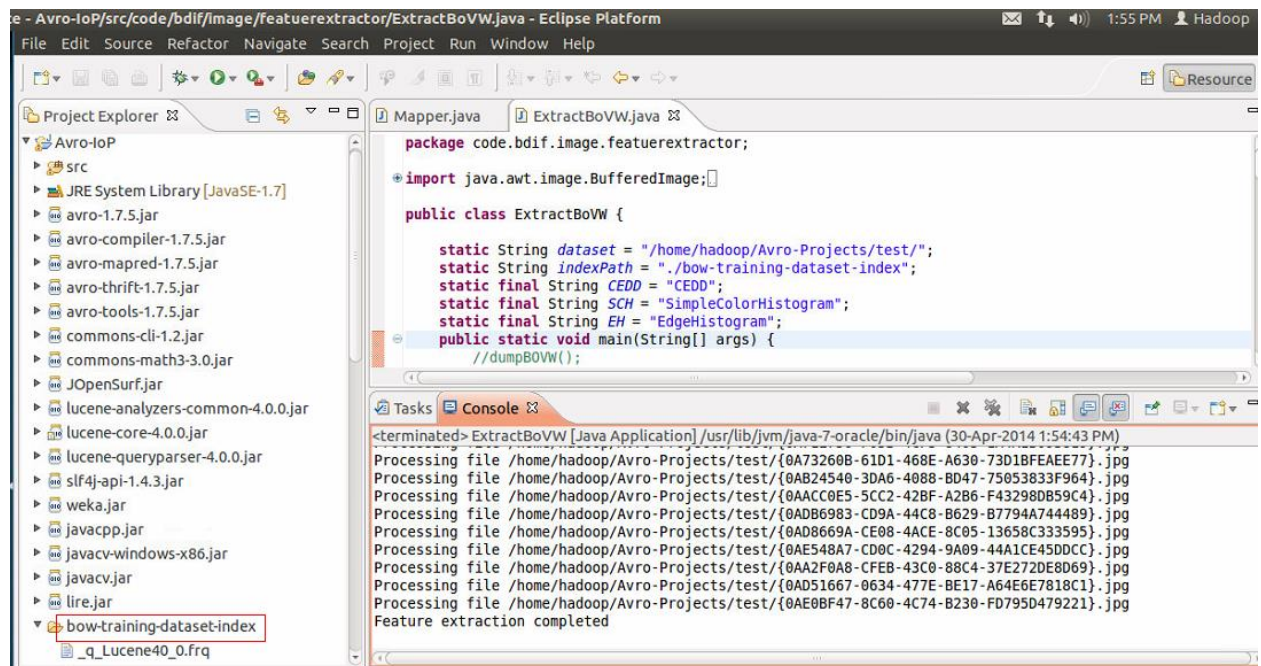


Figure 13-2: Screenshot showing the Feature extraction result



### 13.2. Image Concept Classification Using BDIF Version 1.0

The visual image features generated in LIRE were serialized using BDIF version 1. To incorporate and use the visual image features in a classifier such as Mahout, the data has to be de-serialized. The user has to run DataFormatInterchangeTool API from eclipse IDE to de-serialize the features are generated in the previous step in LIRE. This can run in two modes i.e. CMD mode and GUI mode. If the user does not provide proper arguments to this tool, it won't allow the user proceed further and displays appropriate error and debug messages. To pass arguments in eclipse IDE, the user will utilize the arguments in Run Configurations of the specific class file. The user should ensure that the correct Java Application in the Run Configuration is selected.

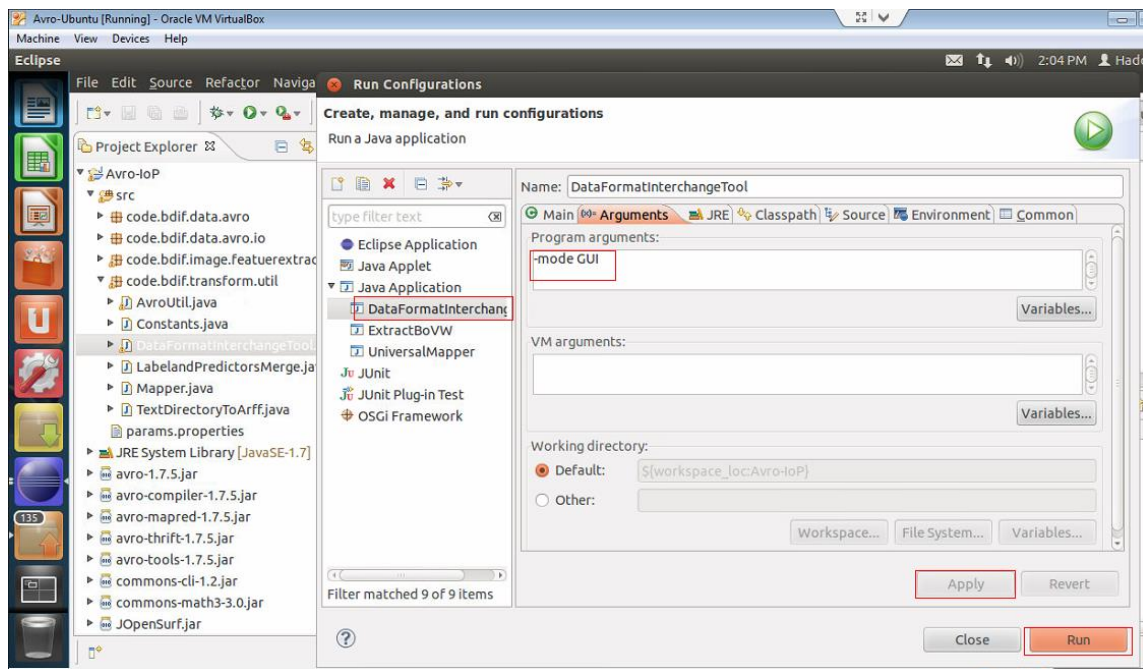


Figure 13-3 – DataFormatInterchangeTool class arguments



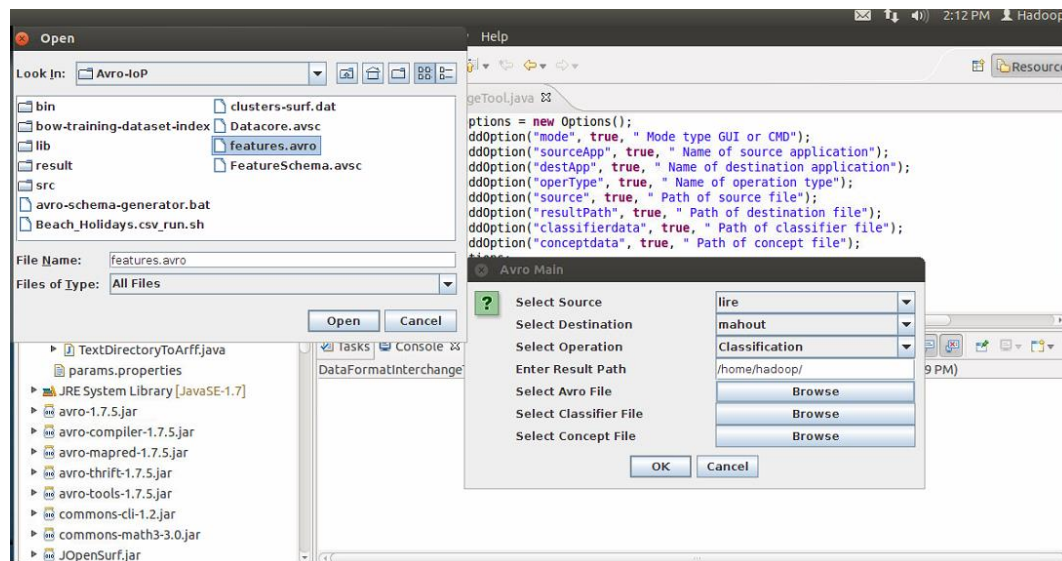


Figure 13-4: BDIF interoperability in GUI Mode

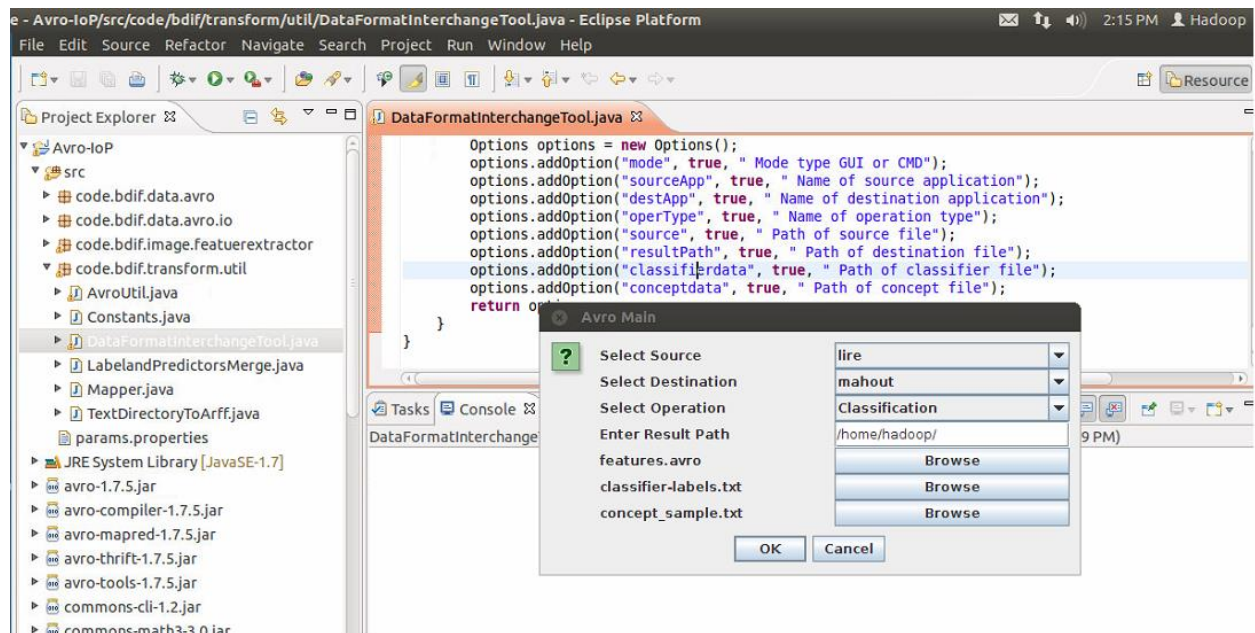


Figure 13-5: Passing arguments in BDIF through UI

After successful deserialization with BDIF the user should find a folder called “result” in the project root directory. (use the refresh function if needed to view the latest result folder). In this result folder the user should find the deserialized.csv and .sh files that is used by Mahout as inputs to the classifier tools.

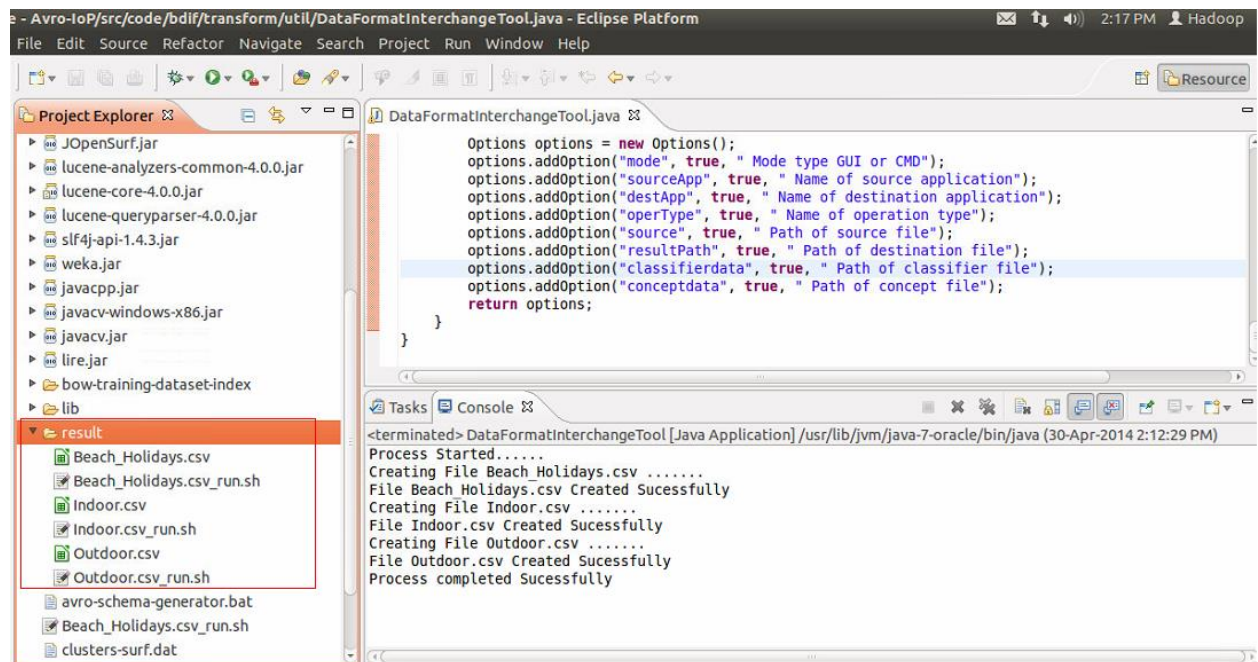


Figure 13-6: Execution log – Result files

The user should set executable permissions for all .sh files and run one of the script files to run the Mahout classifier. This will generate a classifier confusion matrix that details the error rates and correct classification rates for the test images used for visual concept classification in the VCDT task.





### 13.3. Image Concept Classification Using BDAS Spark

To classify the extracted visual features in BDAS Spark, open the scala IDE and execute svmtest.scala. Make sure the the correct file name is provided. The user should ensure that “BDAS” is specified as part of **destApp** argument during the classification process.

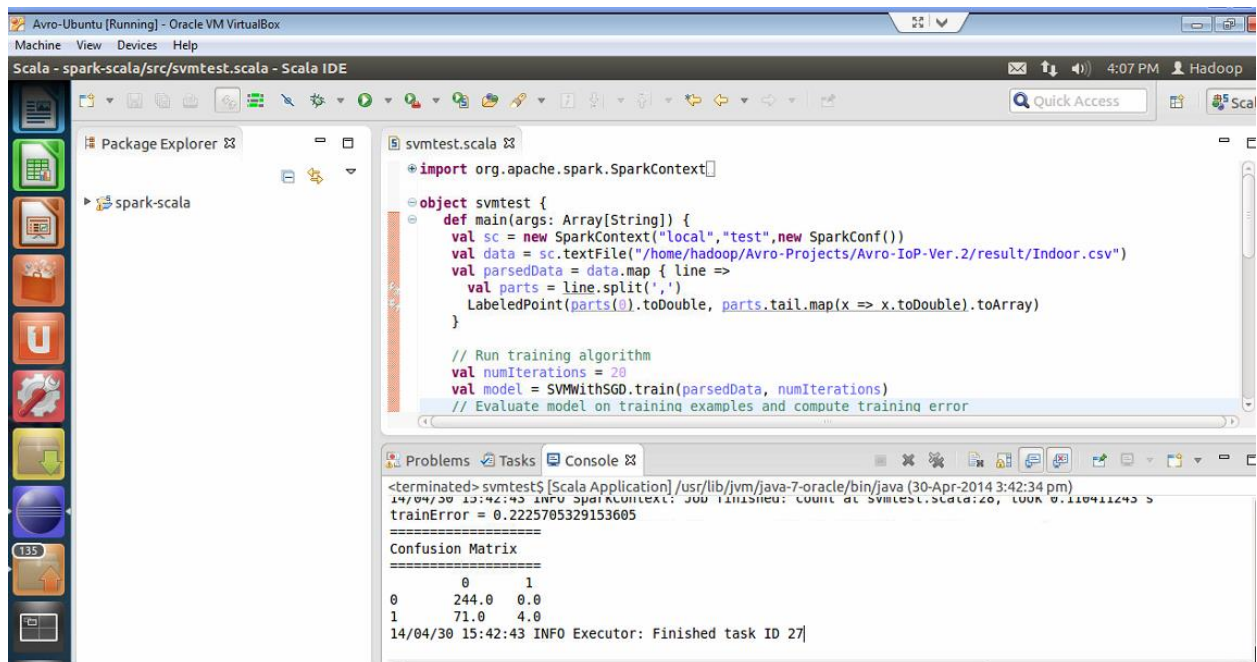


Figure 9: Confusion Matrix in BDAS Spark – Scala IDE