A

Project Report

on

FINDING PERCENTAGE DUPLICATES USING HADOOP BY LEVENSHTEIN DISTANCE MODEL

submitted in partial fulfillment of the requirements

for the award of the degree of

BACHELLOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

K. Santhi Sree (12501A0593)

V. Bhavesh (12501A05B5)

S. Maha Lakshmi (12501A0591)

V.M.N.S. Balagi (12501A05B3)

under the esteemed guidance of

Mr. P. ANIL KUMAR, M.Tech

Asst. Professor, Department of CSE.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PRASAD V POTLURI SIDDHARTHA INSTITUTE OF TECHNOLOGY

(Permanently affiliated to JNTU :: Kakinada, Approved by AICTE)

(An NBA & NAAC accredited and ISO 9001:2008 certified institution)

Kanuru, Vijayawada - 520007

2015-2016.

PRASAD V POTLURI

SIDDHARTHA INSTITUTE OF TECHNOLOGY

(Permanently affiliated to JNTU :: Kakinada, Approved by AICTE)

(An NBA & NAAC accredited and ISO 9001:2008 certified institution)

Kanuru, Vijayawada – 520007

<u>CERTIFICATE</u>

This is to certify that the project report titled **"FINDING PERCENTAGE DUPLICATES USING HADOOP BY LEVENSHTEIN DISTANCE MODEL"** is the bonafide work of

K. Santhi Sree (12501A0593)

V.Bhavesh (12501A05B5),

S. Maha Lakshmi(12501A0591)

V.M.N.S. Balagi (12501A05B3)

in partial fulfillment of the requirements for the award of the graduate degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** during the academic year 2015-2016.

Signature of the Guide | **Signature of the HOD**

Mr. P.Anil Kumar, M.Tech | **Dr. M.V. Rama Krishna, Ph.D**

Department of C.S.E. | **Department of C.S.E.**

Signature of the External

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without a mention of the people, who made it possible and whose guidance and encouragement crown all the efforts with success.

We reckon to distant to endorse out indebtedness and deep sense of gratitude to our beloved **PrincipalDr. K. SivajiBabu**, **M.Tech, Ph.D**, for his encouragement in all endeavours.

We would like to express our profound thanks to **Dr**. **M.V.Rama Krishna, M.Tech, Ph.D, Professor and Head of the Department of Computer Science and Engineering** for his valuable suggestions and guidance in making our project successful.

We are greatly indebted to our project guide **Mr. P. Anil Kumar, M.Tech, Assistant Professor in Department of CSE,** for his patience and cheerful readiness to help us in our project.

We sincerely thank **Dr. A. Sudhir Babu, M.Tech, Ph.D, Professor** for their support, encouragement and suggestions they had given during the course of the project which made us to see the silver line in every dark cloud. It gives us immense pleasure to thank our project coordinators **Mrs.G.LalithaKumari, M.Tech, Assistant Professor and Mrs. J.Rama Devi, M.Tech, Assistant Professor** who have been encouraging and kind enough to help us at times of need.

We are very much grateful to all our **staff and faculty of Department of CSE** for their cooperation during the course of the project work. Our profound thanks for their helping providing the necessary facilities to do the project work. Finally, we would like to express our sincere thanks to each and everyone of our college who have contributed our help and guidance for successful completion of this project.

Project Associates:

K. SanthiSree

V. Bhavesh

S. Maha Lakshmi

V.M.N.S. Balagi

ABSTRACT

The project entitled "Finding percentage duplicates using Big Data by Levenshtein Distance model" aims at string matching technique using Levenshtein distance model. The project has been developed with Big Data as the Front End and Java as the Back End. The project is prepared to implement the code in Java platform using Linux OS.

In Google, the string what we type in text box given must match with the string that is already stored in the database using percentage calculation . As a result numerous results are found in fraction on seconds(1,00,00,000 found in 0.57 seconds). Similarly in Amazon,Flipkart,Snapdeal recommended products are found. So, by this Levenshtein distance model these results are found. This model is all about finding the percentage calculation between two strings using hadoop. This model is the proven model of all the models.

Hadoop performs two tasks: mapper and reducer. The mapper divides the given string into various parts and store it in the local disk and then passes those outputs to the comparator class to group up similar strings and then pass those outputs to the reducer and reducer class merge those outputs and store in the HDFS(Hadoop Distributed File System). We need a parallel processing computing so hadoop is chosen.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

1. INTRODUCTION

1.1 HADOOP DEFINITION

Hadoop is a free, Java-based programming framework that supports the processing of large data sets in a distributed computing environment. It is part of the Apache project sponsored by the Apache Software Foundation.

Hadoop performs two tasks: mapper and reducer. The mapper divides the given string into various parts and store it in the local disk. Then the mapper sends those strings to the comparator to group up similar strings and finally it passes to the reducer to store in Hadoop Distributed File System.

1.2 THE LEVENSHTEIN DISTANCE MODEL

This model is used to find out the distance between two strings using percentage calculation. This model is the proven model of all the models. There are three types of edits namely insert, delete, replace. Based on these edits percentage is calculated.

1.3 PROJECT OVERVIEW

Our system takes the input in simple txt file and groups the similar strings and writes these strings in output txt file. It calculates similarity by calculating number of edits taken to convert one string to another. It operates in terms of insert, delete, replace (3 types of edits) operations. Each edit has a single count. Our system has mainly 4 classes:

1. driver class: connections are given.
2. mapper: passes the strings to be evaluated.
3. comparator: groups similar strings.
4. reducer: writes output to hadoop file system.

In Google, the string what we type in text box given must match with the string that is already stored in the database using percentage calculation . As a result numerous results are found in fraction on seconds(1,00,00,000 found in 0.57 seconds). Similarly in Amazon, Flipkart, Snapdeal recommended products are found. So, by this Levenshtein distance model these results are found. This model is all about finding the percentagecalculation between two strings using hadoop. This model is the proven model of all the models.

1.4 PROPOSED SYSTEM

1. Used in amazon for similar products.
2. We see “found 2k results in 0.3 seconds” in google search engine.
3. It is one of the algorithms used by google search engine.
4. Used in speech recognition and predictable dictionary.
5. Spell checkers.

SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

2.1 STARTING IN ECLIPSE:

- We choose eclipse because it takes care of syntax of program and also it has a debugger.
- We need not worry about the libraries we add as eclipse takes care of it.
- We need to provide input and output to the program through runconfigurations ->arguments.
- We need to provide hadoop libraries to our program.

2.2 IMPLEMENTING A DRIVER

- The driver class defines the connections of mapper, reducer and the comparator.
- Here we define input and output paths for the program.
- We define the connections through writing our mapper, reducer, comparator and driver class names.
- Any error in these connections will lead to the failure of the program.

2.3 IMPLEMENTING A MAPPER

- In this module the given string is divided into various inputs(parts) and store those inputs in the local disk and then produces outputs.
- Here we pass input as key-value pairs where the key is line offset and value is the string passed for evaluation.
- Here in mapper we remove the bad data and send only valid data to comparator.
- We send only the required data that is to be evaluated and neglect all other data

  Syntax:

  ```
  public class dedupMapper extends Mapper<LongWritable,Text,Text,Text>
  {
  ```

```
        public void map(LongWritable key, Text value, Context context)
        {………………….}}
```

2.4 IMPLEMENTING A COMPARATOR

- The main objective of this class is to group up similar strings.
- The output from mapper is passed to the comparator class and comparator matches the strings.
- Here the comparator class which we define is not the default comparator class but our custom class.
- So to use the default comparator class known as writable comparator we use extends.

Syntax:

```
public class groupingComparator1 extends WritableComparator
{
        public groupingComparator1()
        {
                super(Text.class, true);
        }……………
}
```

2.5 COMPUTING THE STRINGS

		C	O	H	E	N
	0	1	2	3	4	5
M	1	1	2	3	4	5
C	2	1	2	3	4	5
C	3	2	2	3	4	5
O	4	3	2	3	4	5
H	5	4	3	2	3	4
N	6	5	4	3	3	3

The Levenshtein distance between two strings is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other.

2.6 IMPLEMENTING A REDUCER

- The output from the comparator class is the input for the reducer.
- Now these inputs are merged and then stored in the HDFS(Hadoop Distributed File System).
- Here we take an array-list to write the output. Since we get multiple output values.
- In reducer we can also have conditions, to evaluate the data directly passed from mapper to reducer.

Syntax:

```
public class dedupReducer extends Reducer<Text,Text,Text,Text>
{
        public void reduce(Text key, Iterable<Text>value,Context context)
        throws IOException, InterruptedException
        {
        --------------------
        }
        --------
 }
```

2.7 RUNNING THE PROGRAM

- Execute the command.
- Hadoop jar filename.jar input-directory output-directory.
- We should not give the output file name that already exists because hadoop will not override, instead it shows error as filename already exists failing job.

2.8ACCESSING THE OUTPUT

- In hadoop the output files which are stored in hdfs(hadoop distributed file system) are not generally visible.
- We can see the output in terminal by either using "cat" command or hadoop provides a html link as "localhost:50070" which has hdfs file browser.
- Through this browser we download the output data.
- The output data is generally a ".txt" file.

FEASIBILITY STUDY

3. FEASIBILITY STUDY

A feasibility analysis usually involves a through assessment of the operational (need), financial and technical aspects of a proposal. Feasibility study is the test of the system proposal made to identify whether the user needs may be satisfied using the current software and hardware technologies, whether the system will be cost effective from a business point of view and whether it can be developed with the given budgetary constraints. A feasibility study should be relatively cheap and done at the earliest possible time. Depending on the study, the decision is made whether to go head with a more detailed analysis. When a new project is proposed, it normally goes through feasibility assessment. Feasibility study is carried out to determine whether the proposed system is possible to develop with available resources and what should be the cost consideration. Facts considered in the feasibility analysis were:

3.1 TECHNICAL FEASIBILITY

Technical feasibility includes whether the technology is available in the market for development and its availability. The assessment of technical feasibility must be based on an outline design of system requirements in terms of input, output, files, programs and procedures. This can be qualified in terms of volumes of data, trends, frequency of updating, cycles of activity etc, in order to give an introduction of technical system. Considering our project it is technical feasible. Training and Placement System, with its emphasis on a more strategic decision making process is fast gaining ground as a popular outsourced function.

3.2 OPERATIONAL FEASIBILITY

This analysis involves how it will work when it is installed and the assessment of managerial environment in which it is implemented. People are inherently resistant to change and computers have been known to facilitate change. The new proposed system is very much useful to the users and therefore it will accept broad audience from around the world.

3.3ECONOMIC FEASIBILITY

This feasibility study present tangible and intangible benefits from the project by comparing the development and operational cost. The technique of cost benefit analysis is often used as a basis for assessing economic feasibility. This system needs some more initial investment than the existing system, but it can be justifiable that it will improve quality of service. Thus feasibility study should center along the following points:

- Improvement resulting over the existing method in terms of accuracy, timeliness.
- Cost comparison
- Estimate on the life expectancy of the hardware.

SYSTEM REQUIREMENTS

4. SYSTEM REQUIREMENTS

4.1 FUNCTIONAL REQUIREMENTS

The main purpose of functional requirements within the requirement specification document is to define all the activities or operations that take place in the system. These are derived through interactions with the users of the system. Since the Requirements Specification is a comprehensive document & contains a lot of data, it has been broken down into different Chapters in this report. The depiction of the Design of the System in UML is presented in a separate chapter.

4.2 NON-FUNCTIONAL REQUIREMENTS

The non-functional requirements consist of

4.2.1 Analysis, Design & Data requirements (Use-case diagrams, textual analysis, sequence diagrams,deployment diagrams etc.).

4.2.2 Validation Criteria.

4.2.1 ANALYSIS, DESIGN& DATA REQIREMENTS

- The use case diagrams, textual analysis and sequence diagrams fall into this category. Since each category above is of considerable importance, they have been dealt in separate chapters. An outline is only included here.
- The Analysis & Design phases of the system yield Use Case Diagram, Deployment Diagram, Logical Diagram and Activity Diagram.

4.2.2 VALIDATION CRITERIA

- The Validation Criteria are dealt separately in the Chapter dealing with the Test Strategy & Test cases.

4.3 Hardware Requirements:

• Processor	:	Pentium
• RAM	:	1 GB
• Hardisk	:	depends on data to be processed (recommended 40gb minimum)

4.4 Software Requirements:

• Operating System	:	Linux, Ubuntu for windows
• Front-End	:	Hadoop
• Back-End	:	Java
• Other software	:	Eclipse (version 2.3 and above)

SYSTEM DESIGN

5. SYSTEM DESIGN

5.1 INTRODUCTION TO SYSTEM DESIGN:

Design is concerned with identifying software components and specifying relationships among components. It is used to provide software structure and also provides a blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts, in such a manner, that the interaction between parts is minimal and clearly specified. Design will explain software components in detail. This will help the implementation of the system. Moreover, this will guide for further changes in the system to satisfy the future requirements.

5.1.1 INPUT DESIGN:

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

5.1.2 OUTPUT DESIGN:

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user.

5.2 USECASE DIAGRAM:

A Use Case Diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis.The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

USE CASE DIAGRAM COMMONLY CONTAIN

- Use Cases
- Actors
- Dependency
- Generalization
- Association relationships

Use cases:

A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

Actors:

An actor is a person, organization, or external system that plays a role in one or more interactions with the system.

System boundary boxes (optional):

A rectangle is drawn around the use cases, called the system boundary box, to indicate the scope of system. Anything within the box represents functionality that is in scope and anything outside the box is not. Four relationships among use cases are used often in practice.

Include:

In one form of interaction, a given use case may include another. "Include is a Directed Relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case.The first use case often depends on the outcome of the included use case. This is useful for extracting truly common behaviors from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use case, with the label "«include»".There are no parameters or return values. To specify the location in a flow of events in which the base use case includes the behavior of another, you simply write include followed by the name of use case you want to include, as in the following flow for track order.

Extend:

In another form of interaction, a given use case (the extension) may extend another. This relationship indicates that the behavior of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label "«extend»". Modelers use the «extend» relationship to indicate use cases that are "optional" to the base use case.

Generalization:

In the third form of relationship among use cases, a generalization/specialization relationship exists. A given use case may have common behaviors, requirements, constraints, and assumptions with a more general use case. In this case, describe them once, and deal with it in the same way, describing any differences in the specialized cases. The notation is a solid line ending in a hollow triangle drawn from the specialized to the more general use case (following the standard generalization notation.

Associations:

Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modeled as lines connecting use cases and actors to one another, with an optionalarrowhead on one end of the line. The arrowhead is often used to indicating the direction of the initial invocation of the relationship or to indicate the primary actor within the use case.

Identified Use Cases:

The "user model view"encompasses a problem and solution from the preservative of those individuals whose problem the solution addresses. The view presents the goals and objectives of the problem owners and their requirements of the solution. This view is composed of "use case diagrams". These diagrams describe the functionality provided by a system to external integrators. These diagrams contain actors, use cases, and their relationships.

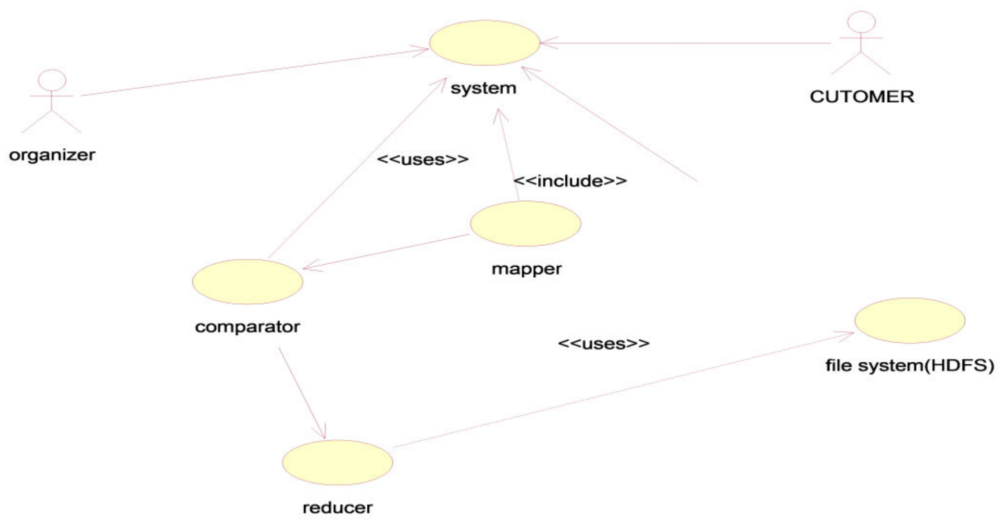


Figure:1

Use case Diagrams represent the functionality of the system from a user's point of view.
Figure:1 are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.

5.3 DEPLOYMENT DIAGRAM

Figure:2 represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware used to deploy the application.

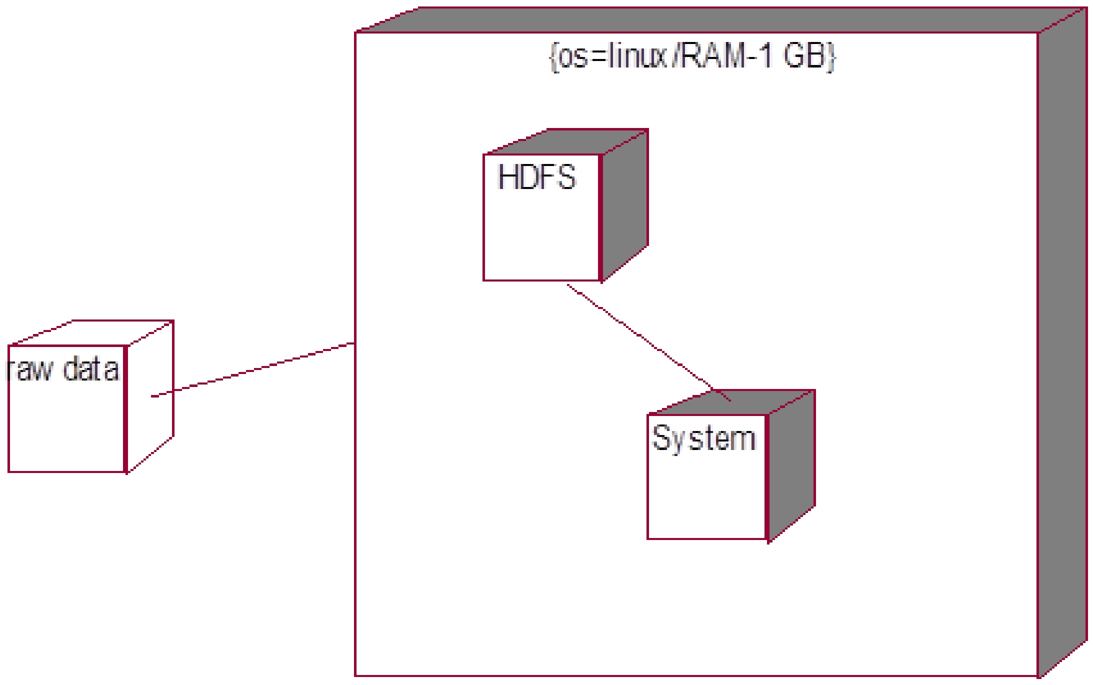


Figure : 2

5.4 LOGICAL DIAGRAM

Figure:3represents a static view of the objects and classes that make up the design/analysis space. Typically, a Domain Model is a looser, high level view of Business Objects and entities, while the Class Model is a more rigorous and design focused model. This discussion relates mainly to the Class Model.

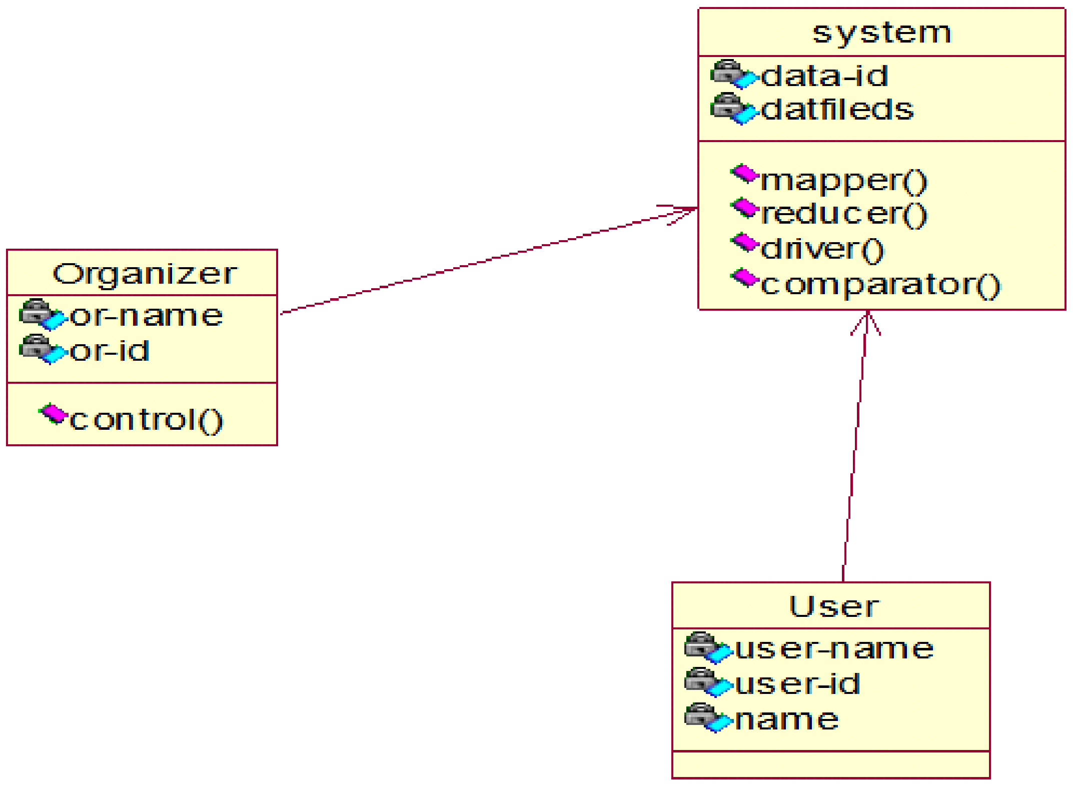


Figure-3

5.5 ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. **Figure:4** is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system.So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.

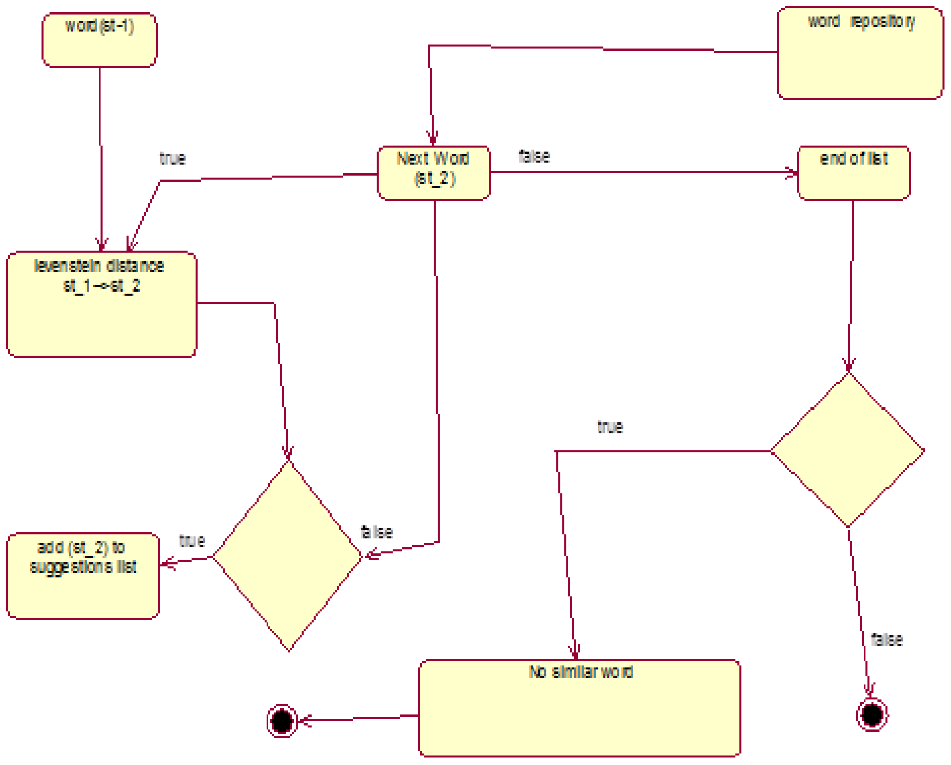


Figure-4

SYSTEM IMPLEMENTATION

6. SYSTEM IMPLEMENTATION

6.1 ABOUT HADOOP:

Hadoop is a free, Java-based programming framework that supports the processing of large data sets in a distributed computing environment. It is part of the Apache project sponsored by the Apache Software Foundation.

Hadoop performs two tasks: mapper and reducer. The mapper divides the given string into various parts and store it in the local disk. Then the mapper sends those strings to the comparator to group up similar strings and finally it passes to the reducer to store in Hadoop Distributed File System.

Our system takes the input in simple txt file and groups the similar strings and writes these strings in output txt file. It calculates similarity by calculating number of edits taken to convert one string to another. It operates in terms of insert, delete, replace (3 types of edits) operations. Each edit has a single count. Our system has mainly 4 classes:

1. driver class: connections are given.
2. mapper: passes the strings to be evaluated.
3. comparator: groups similar strings.
4. reducer: writes output to hadoop file system

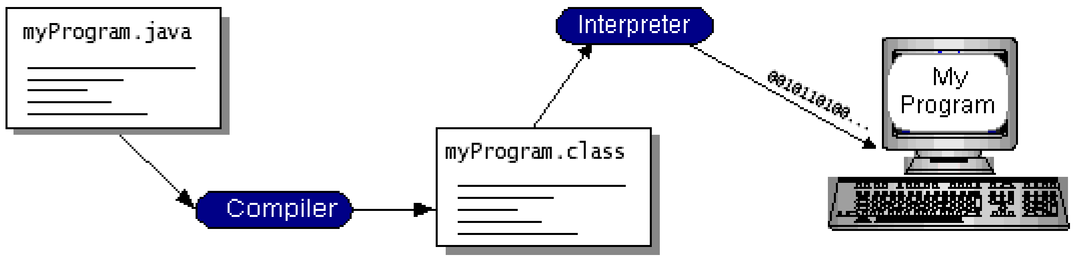
6.2ABOUT JAVA

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

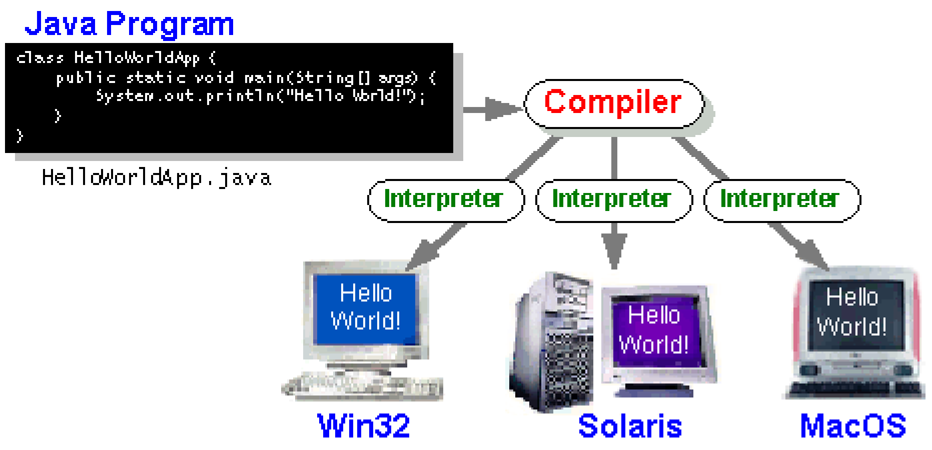
- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded

- Robust
- Dynamic
- Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes* —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



You can think of Java byte codes as the machine code instructions for the *Java Virtual Machine* (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

THE JAVA PLATFORM

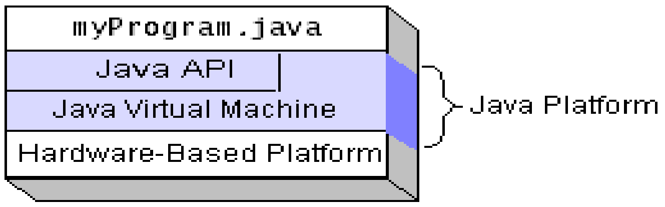
A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

- The *Java Virtual Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms. The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

What Can Java Technology Do?

The most common types of programs written in the Java programming language are *applets* and *applications*. If you've surfed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

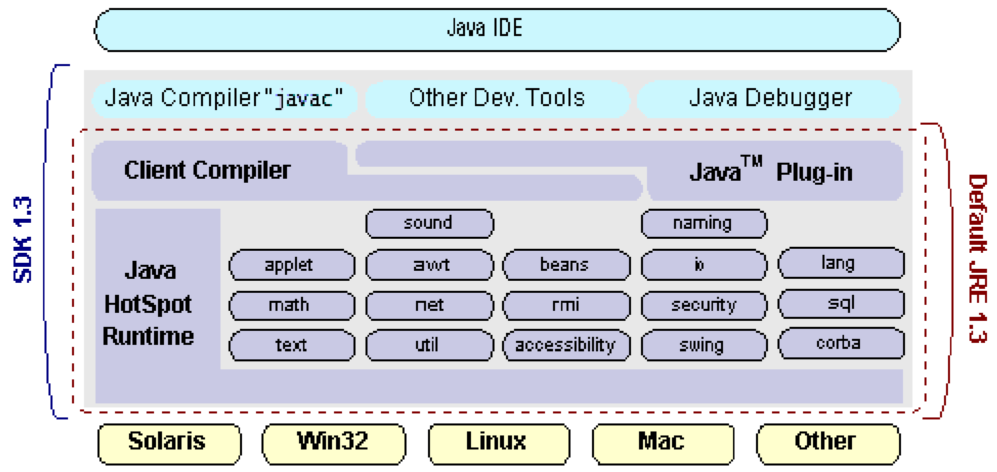
However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a *server* serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a *servlet*. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

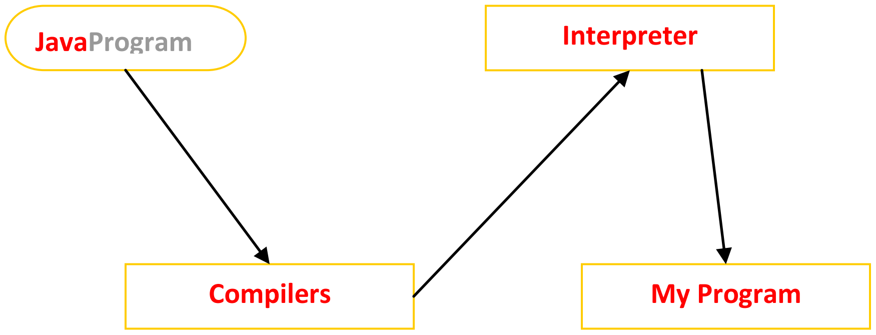
- **The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
- **Applets**: The set of conventions used by applets.
- **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
- **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
- **Software components**: Known as JavaBeans^{TM}, can plug into existing component architectures.
- **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC^{TM})**: Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.



Java is also unusual in that each Java program is both compiled and interpreted. With a compile you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer.

Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this works.



You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

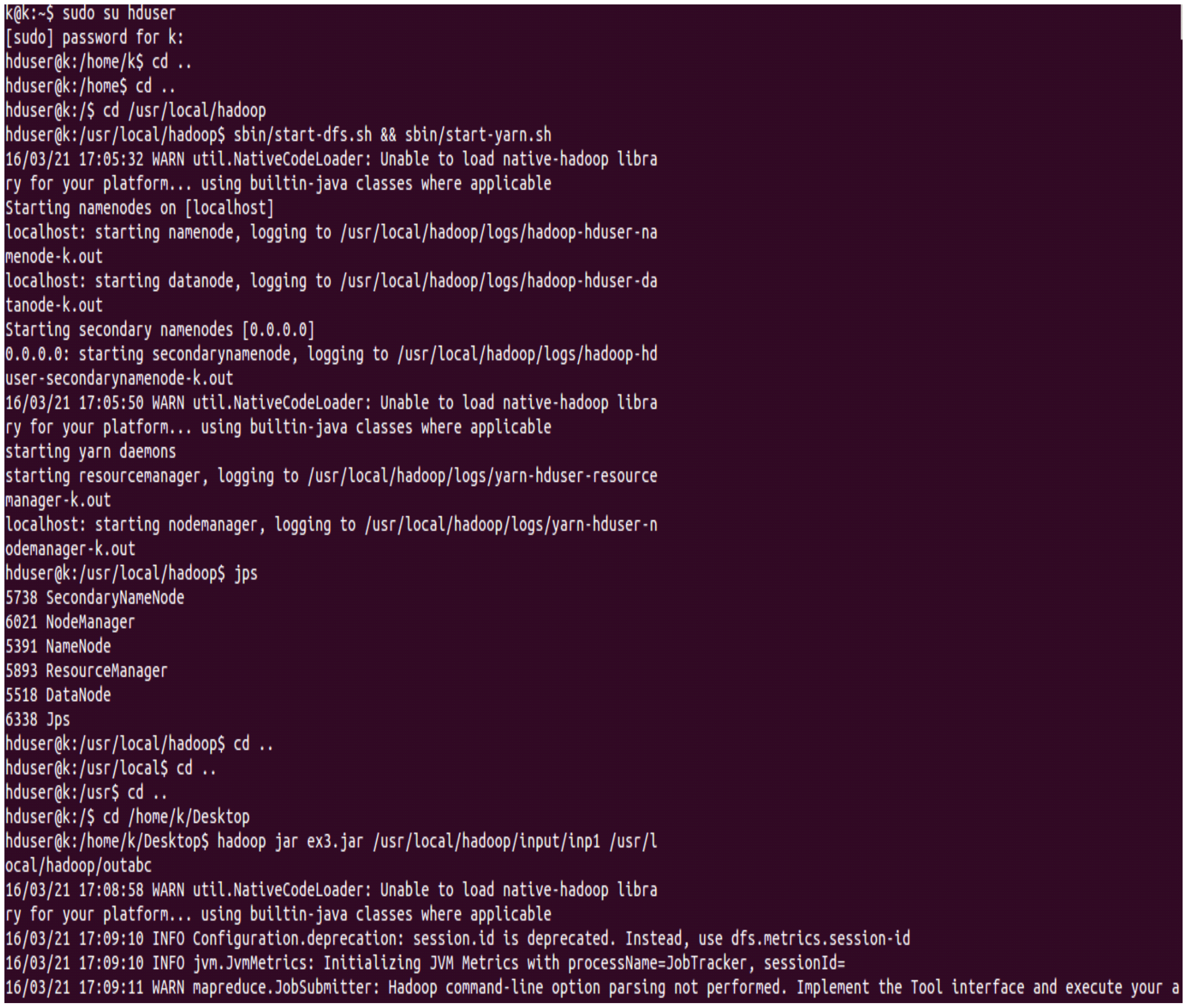
Java byte codes help make "write once, run anywhere" possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh.

SCREEN SHOTS

7. SCREENSHOTS

7.1 HADOOP INSTALLATION

Screenshot-1



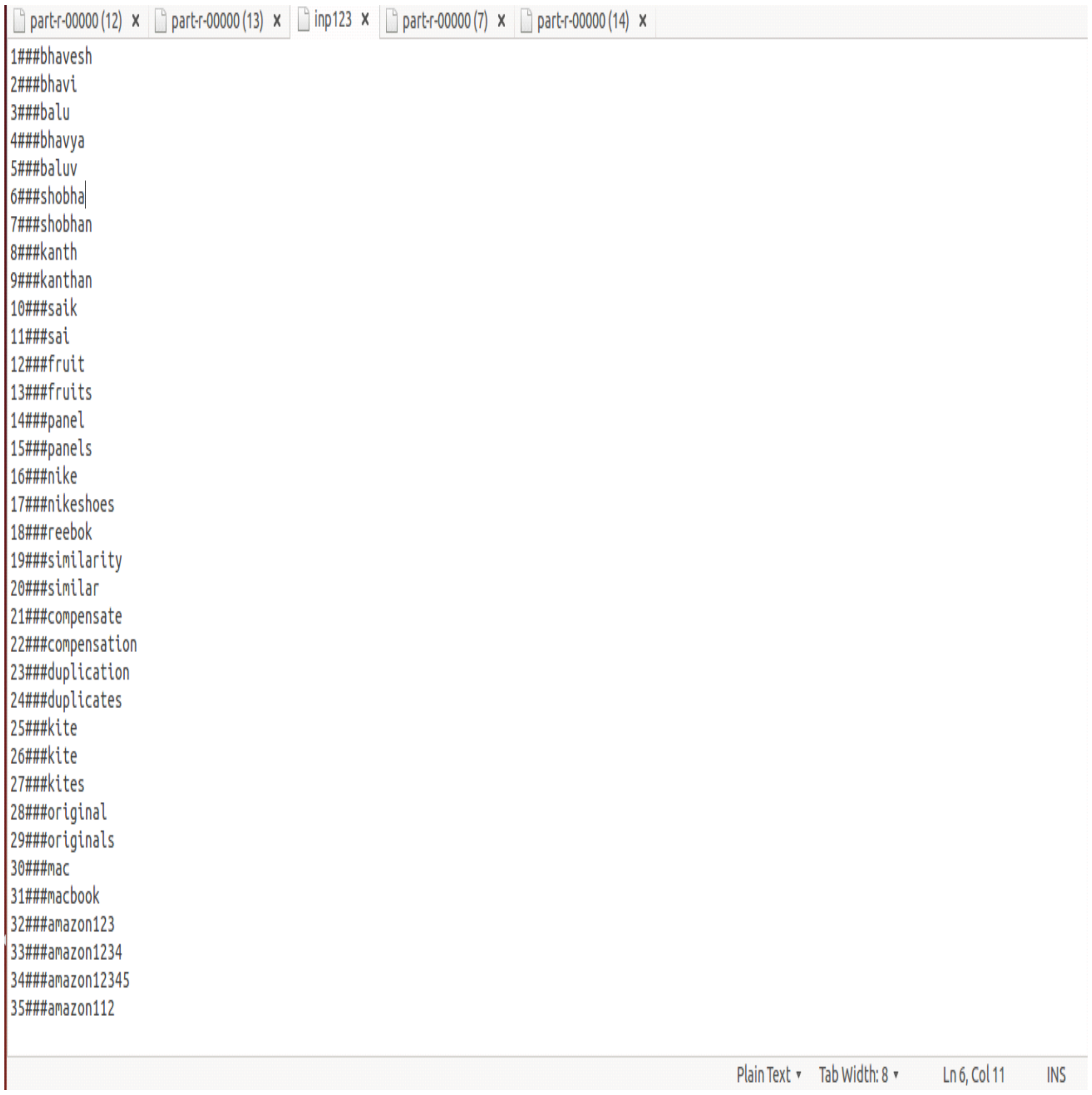
```
k@k:~$ sudo su hduser
[sudo] password for k:
hduser@k:/home/k$ cd ..
hduser@k:/home$ cd ..
hduser@k:/$ cd /usr/local/hadoop
hduser@k:/usr/local/hadoop$ sbin/start-dfs.sh && sbin/start-yarn.sh
16/03/21 17:05:32 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-na
menode-k.out
localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-da
tanode-k.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hd
user-secondarynamenode-k.out
16/03/21 17:05:50 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resource
manager-k.out
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-n
odemanager-k.out
hduser@k:/usr/local/hadoop$ jps
5738 SecondaryNameNode
6021 NodeManager
5391 NameNode
5893 ResourceManager
5518 DataNode
6338 Jps
hduser@k:/usr/local/hadoop$ cd ..
hduser@k:/usr/local$ cd ..
hduser@k:/usr$ cd ..
hduser@k:/$ cd /home/k/Desktop
hduser@k:/home/k/Desktop$ hadoop jar ex3.jar /usr/local/hadoop/input/inp1 /usr/l
ocal/hadoop/outabc
16/03/21 17:08:58 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
16/03/21 17:09:10 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
16/03/21 17:09:10 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
16/03/21 17:09:11 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your a
```

First login to the required user and then install hadoop and finally execute the input.

7.2 TEST CASE-1

7.2.1 INPUT DATA

Screenshot-2

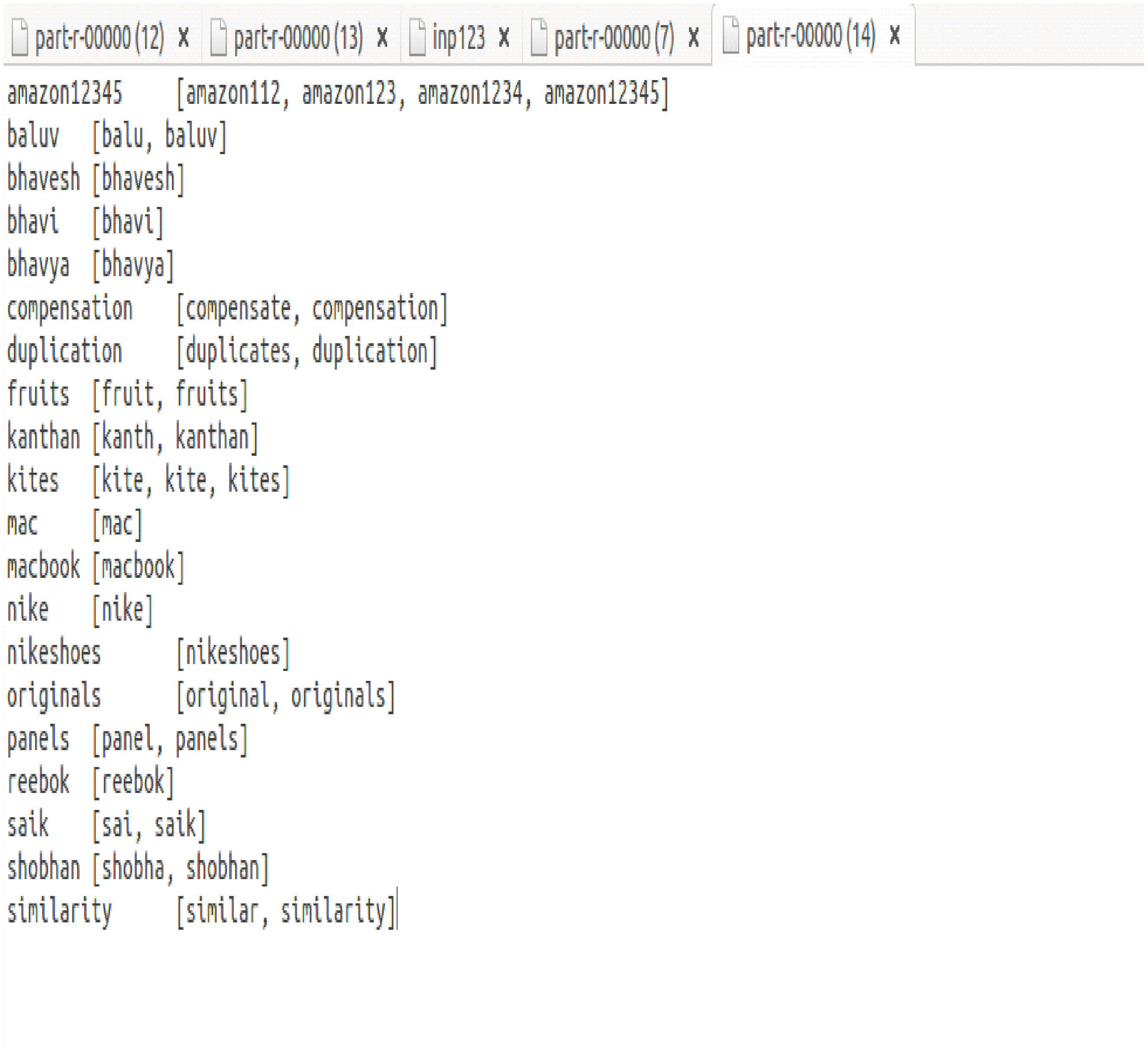


```
1###bhavesh
2###bhavi
3###balu
4###bhavya
5###baluv
6###shobha
7###shobhan
8###kanth
9###kanthan
10###saik
11###sai
12###fruit
13###fruits
14###panel
15###panels
16###nike
17###nikeshoes
18###reebok
19###similarity
20###similar
21###compensate
22###compensation
23###duplication
24###duplicates
25###kite
26###kite
27###kites
28###original
29###originals
30###mac
31###macbook
32###amazon123
33###amazon1234
34###amazon12345
35###amazon112
```

(Raw Data written manually)

7.2.2 OUTPUT DATA

Screenshot-3



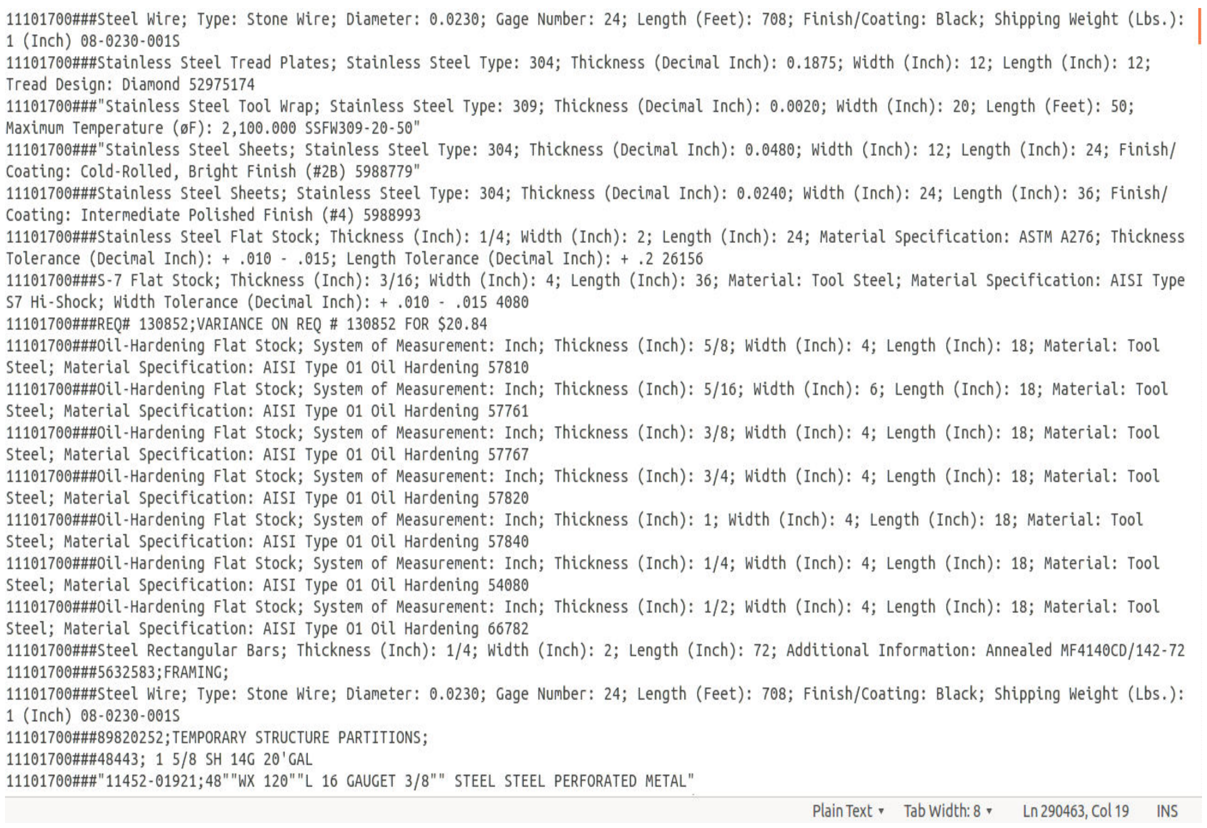
```
amazon12345	[amazon112, amazon123, amazon1234, amazon12345]
baluv	[balu, baluv]
bhavesh [bhavesh]
bhavi	[bhavi]
bhavya	[bhavya]
compensation	[compensate, compensation]
duplication	[duplicates, duplication]
fruits	[fruit, fruits]
kanthan [kanth, kanthan]
kites	[kite, kite, kites]
mac	[mac]
macbook [macbook]
nike	[nike]
nikeshoes	[nikeshoes]
originals	[original, originals]
panels	[panel, panels]
reebok	[reebok]
saik	[sai, saik]
shobhan [shobha, shobhan]
similarity	[similar, similarity]
```

(strings are compared and stored as key-value pairs)

7.3 TEST CASE-2

7.3.1 INPUT DATA

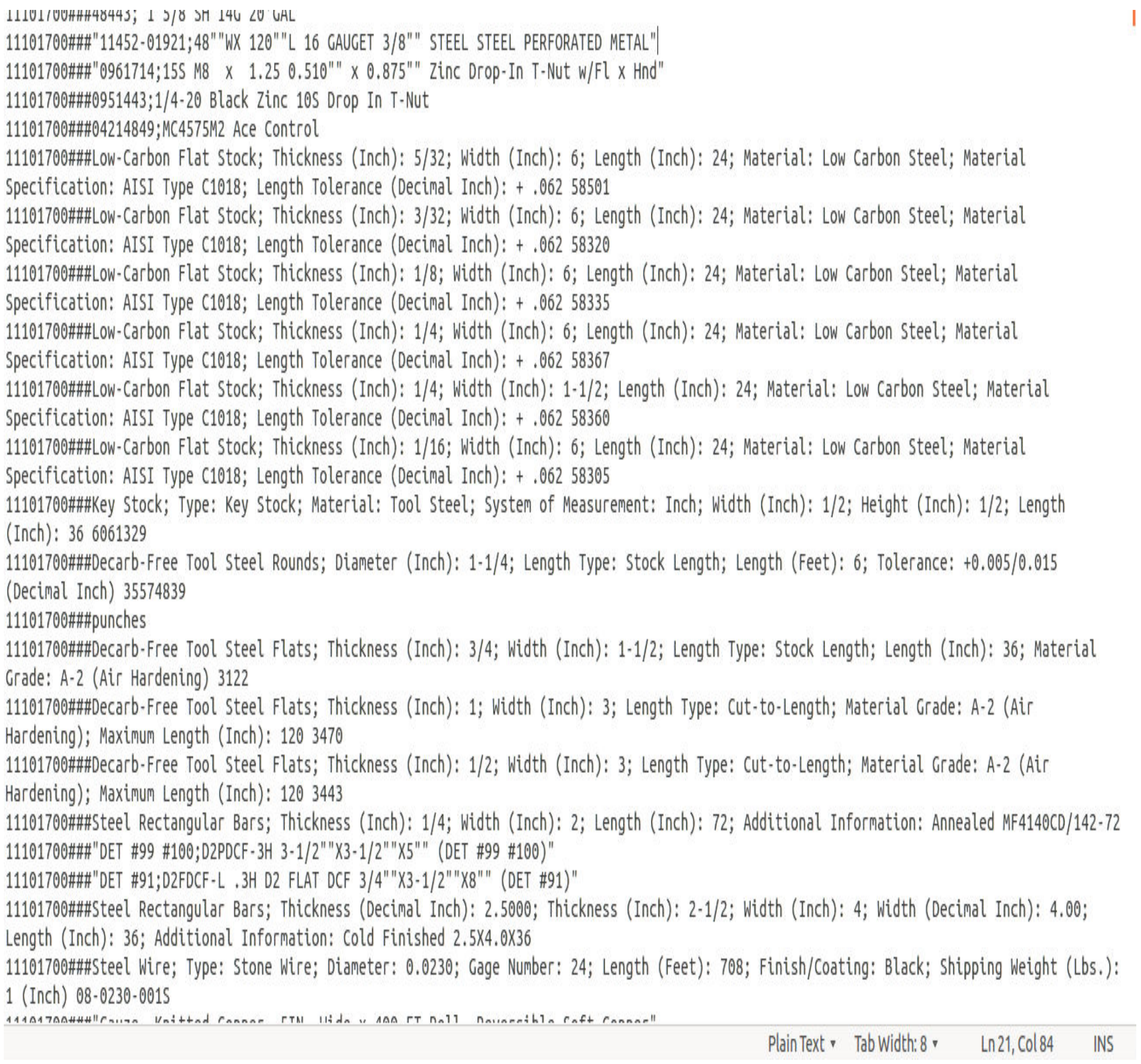
Screenshot-4



```
11101700###Steel Wire; Type: Stone Wire; Diameter: 0.0230; Gage Number: 24; Length (Feet): 708; Finish/Coating: Black; Shipping Weight (Lbs.):
1 (Inch) 08-0230-001S
11101700###Stainless Steel Tread Plates; Stainless Steel Type: 304; Thickness (Decimal Inch): 0.1875; Width (Inch): 12; Length (Inch): 12;
Tread Design: Diamond 52975174
11101700###"Stainless Steel Tool Wrap; Stainless Steel Type: 309; Thickness (Decimal Inch): 0.0020; Width (Inch): 20; Length (Feet): 50;
Maximum Temperature (øF): 2,100.000 SSFW309-20-50"
11101700###"Stainless Steel Sheets; Stainless Steel Type: 304; Thickness (Decimal Inch): 0.0480; Width (Inch): 12; Length (Inch): 24; Finish/
Coating: Cold-Rolled, Bright Finish (#2B) 5988779"
11101700###Stainless Steel Sheets; Stainless Steel Type: 304; Thickness (Decimal Inch): 0.0240; Width (Inch): 24; Length (Inch): 36; Finish/
Coating: Intermediate Polished Finish (#4) 5988993
11101700###Stainless Steel Flat Stock; Thickness (Inch): 1/4; Width (Inch): 2; Length (Inch): 24; Material Specification: ASTM A276; Thickness
Tolerance (Decimal Inch): + .010 - .015; Length Tolerance (Decimal Inch): + .2 26156
11101700###S-7 Flat Stock; Thickness (Inch): 3/16; Width (Inch): 4; Length (Inch): 36; Material: Tool Steel; Material Specification: AISI Type
S7 Hi-Shock; Width Tolerance (Decimal Inch): + .010 - .015 4080
11101700###REQ# 130852;VARIANCE ON REQ # 130852 FOR $20.84
11101700###Oil-Hardening Flat Stock; System of Measurement: Inch; Thickness (Inch): 5/8; Width (Inch): 4; Length (Inch): 18; Material: Tool
Steel; Material Specification: AISI Type O1 Oil Hardening 57810
11101700###Oil-Hardening Flat Stock; System of Measurement: Inch; Thickness (Inch): 5/16; Width (Inch): 6; Length (Inch): 18; Material: Tool
Steel; Material Specification: AISI Type O1 Oil Hardening 57761
11101700###Oil-Hardening Flat Stock; System of Measurement: Inch; Thickness (Inch): 3/8; Width (Inch): 4; Length (Inch): 18; Material: Tool
Steel; Material Specification: AISI Type O1 Oil Hardening 57767
11101700###Oil-Hardening Flat Stock; System of Measurement: Inch; Thickness (Inch): 3/4; Width (Inch): 4; Length (Inch): 18; Material: Tool
Steel; Material Specification: AISI Type O1 Oil Hardening 57820
11101700###Oil-Hardening Flat Stock; System of Measurement: Inch; Thickness (Inch): 1; Width (Inch): 4; Length (Inch): 18; Material: Tool
Steel; Material Specification: AISI Type O1 Oil Hardening 57840
11101700###Oil-Hardening Flat Stock; System of Measurement: Inch; Thickness (Inch): 1/4; Width (Inch): 4; Length (Inch): 18; Material: Tool
Steel; Material Specification: AISI Type O1 Oil Hardening 54080
11101700###Oil-Hardening Flat Stock; System of Measurement: Inch; Thickness (Inch): 1/2; Width (Inch): 4; Length (Inch): 18; Material: Tool
Steel; Material Specification: AISI Type O1 Oil Hardening 66782
11101700###Steel Rectangular Bars; Thickness (Inch): 1/4; Width (Inch): 2; Length (Inch): 72; Additional Information: Annealed MF4140CD/142-72
11101700###5632583;FRAMING;
11101700###Steel Wire; Type: Stone Wire; Diameter: 0.0230; Gage Number: 24; Length (Feet): 708; Finish/Coating: Black; Shipping Weight (Lbs.):
1 (Inch) 08-0230-001S
11101700###89820252;TEMPORARY STRUCTURE PARTITIONS;
11101700###48443; 1 5/8 SH 14G 20'GAL
11101700###"11452-01921;48""WX 120""L 16 GAUGET 3/8"" STEEL STEEL PERFORATED METAL"
```

(Raw Data of a Steel Industry)

Screenshot-5

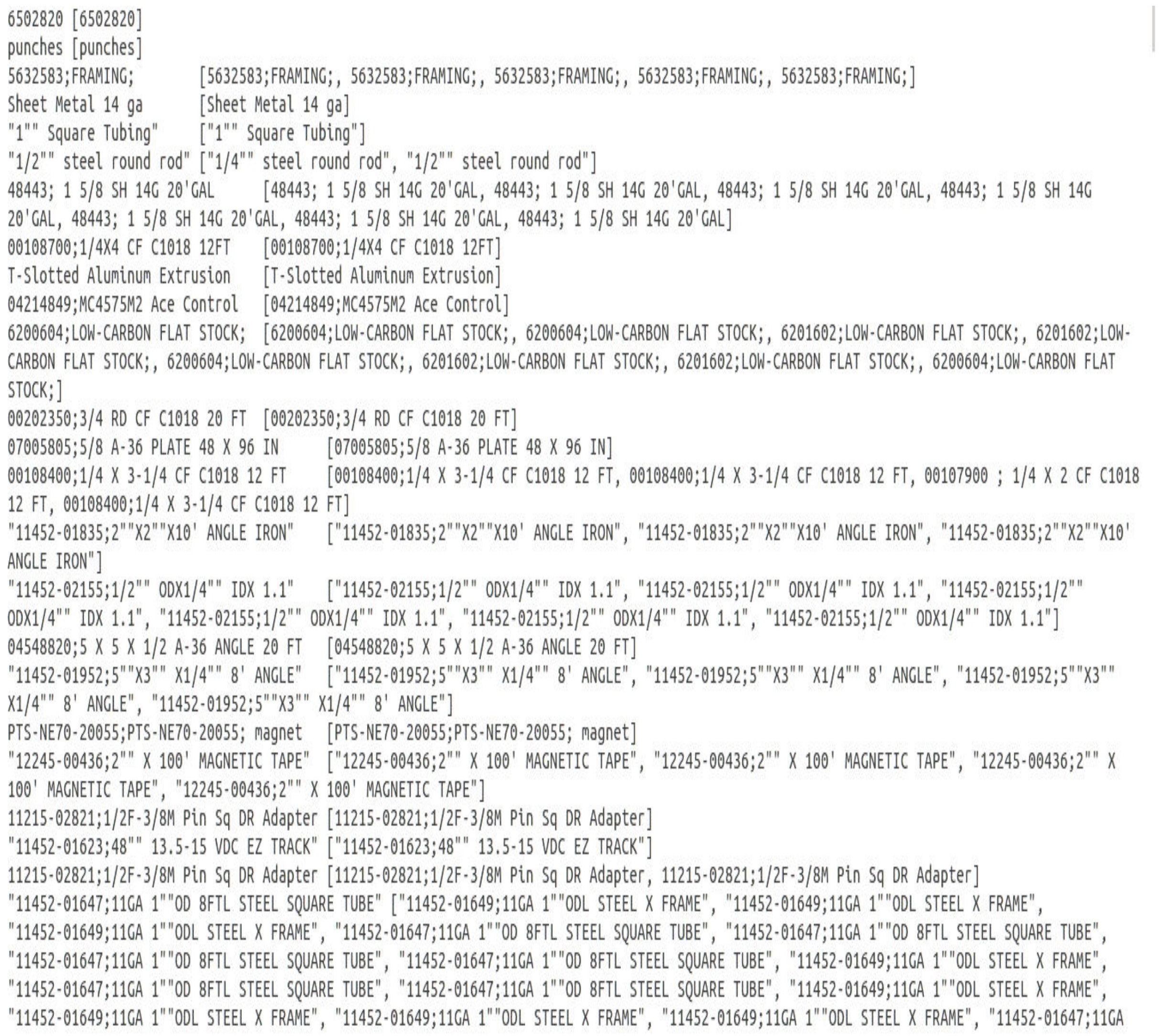


```
11101700###48443; 1 5/8 SM 14G 20 GAL
11101700###"11452-01921;48""WX 120""L 16 GAUGET 3/8"" STEEL STEEL PERFORATED METAL"
11101700###"0961714;15S M8  x  1.25 0.510"" x 0.875"" Zinc Drop-In T-Nut w/Fl x Hnd"
11101700###0951443;1/4-20 Black Zinc 10S Drop In T-Nut
11101700###04214849;MC4575M2 Ace Control
11101700###Low-Carbon Flat Stock; Thickness (Inch): 5/32; Width (Inch): 6; Length (Inch): 24; Material: Low Carbon Steel; Material
Specification: AISI Type C1018; Length Tolerance (Decimal Inch): + .062 58501
11101700###Low-Carbon Flat Stock; Thickness (Inch): 3/32; Width (Inch): 6; Length (Inch): 24; Material: Low Carbon Steel; Material
Specification: AISI Type C1018; Length Tolerance (Decimal Inch): + .062 58320
11101700###Low-Carbon Flat Stock; Thickness (Inch): 1/8; Width (Inch): 6; Length (Inch): 24; Material: Low Carbon Steel; Material
Specification: AISI Type C1018; Length Tolerance (Decimal Inch): + .062 58335
11101700###Low-Carbon Flat Stock; Thickness (Inch): 1/4; Width (Inch): 6; Length (Inch): 24; Material: Low Carbon Steel; Material
Specification: AISI Type C1018; Length Tolerance (Decimal Inch): + .062 58367
11101700###Low-Carbon Flat Stock; Thickness (Inch): 1/4; Width (Inch): 1-1/2; Length (Inch): 24; Material: Low Carbon Steel; Material
Specification: AISI Type C1018; Length Tolerance (Decimal Inch): + .062 58360
11101700###Low-Carbon Flat Stock; Thickness (Inch): 1/16; Width (Inch): 6; Length (Inch): 24; Material: Low Carbon Steel; Material
Specification: AISI Type C1018; Length Tolerance (Decimal Inch): + .062 58305
11101700###Key Stock; Type: Key Stock; Material: Tool Steel; System of Measurement: Inch; Width (Inch): 1/2; Height (Inch): 1/2; Length
(Inch): 36 6061329
11101700###Decarb-Free Tool Steel Rounds; Diameter (Inch): 1-1/4; Length Type: Stock Length; Length (Feet): 6; Tolerance: +0.005/0.015
(Decimal Inch) 35574839
11101700###punches
11101700###Decarb-Free Tool Steel Flats; Thickness (Inch): 3/4; Width (Inch): 1-1/2; Length Type: Stock Length; Length (Inch): 36; Material
Grade: A-2 (Air Hardening) 3122
11101700###Decarb-Free Tool Steel Flats; Thickness (Inch): 1; Width (Inch): 3; Length Type: Cut-to-Length; Material Grade: A-2 (Air
Hardening); Maximum Length (Inch): 120 3470
11101700###Decarb-Free Tool Steel Flats; Thickness (Inch): 1/2; Width (Inch): 3; Length Type: Cut-to-Length; Material Grade: A-2 (Air
Hardening); Maximum Length (Inch): 120 3443
11101700###Steel Rectangular Bars; Thickness (Inch): 1/4; Width (Inch): 2; Length (Inch): 72; Additional Information: Annealed MF4140CD/142-72
11101700###"DET #99 #100;D2PDCF-3H 3-1/2""X3-1/2""X5"" (DET #99 #100)"
11101700###"DET #91;D2FDCF-L .3H D2 FLAT DCF 3/4""X3-1/2""X8"" (DET #91)"
11101700###Steel Rectangular Bars; Thickness (Decimal Inch): 2.5000; Thickness (Inch): 2-1/2; Width (Inch): 4; Width (Decimal Inch): 4.00;
Length (Inch): 36; Additional Information: Cold Finished 2.5X4.0X36
11101700###Steel Wire; Type: Stone Wire; Diameter: 0.0230; Gage Number: 24; Length (Feet): 708; Finish/Coating: Black; Shipping Weight (Lbs.):
1 (Inch) 08-0230-001S
11101700###"Gauze, Knitted Copper, FIN. Wide x 400 FT Roll, Reversible Soft Copper"
```

(continued input of 6.3.1)

7.3.2 OUTPUT DATA

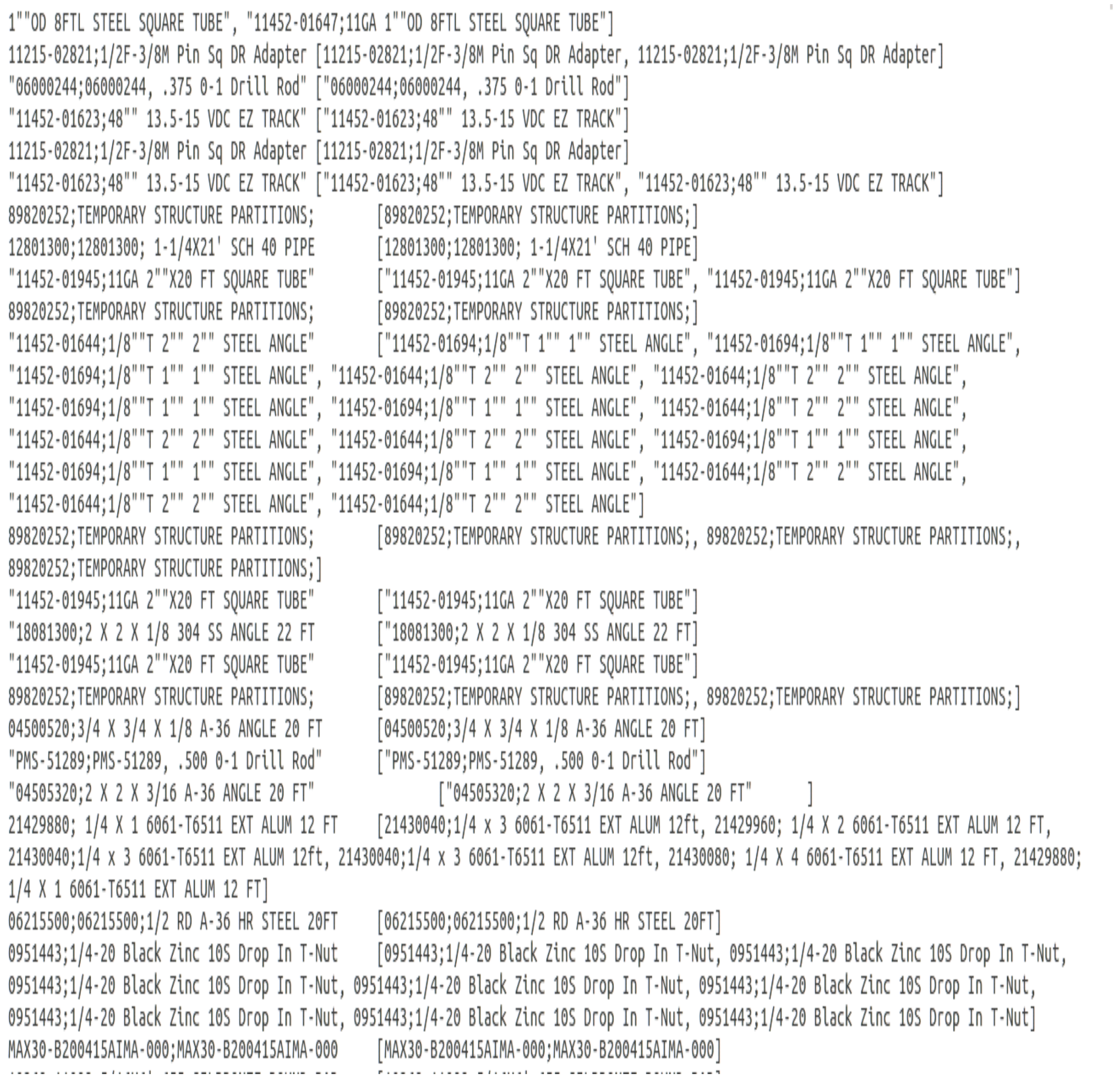
Screenshot-6



```
6502820 [6502820]
punches [punches]
5632583;FRAMING;       [5632583;FRAMING;, 5632583;FRAMING;, 5632583;FRAMING;, 5632583;FRAMING;, 5632583;FRAMING;]
Sheet Metal 14 ga      [Sheet Metal 14 ga]
"1"" Square Tubing"    ["1"" Square Tubing"]
"1/2"" steel round rod" ["1/4"" steel round rod", "1/2"" steel round rod"]
48443; 1 5/8 SH 14G 20'GAL     [48443; 1 5/8 SH 14G 20'GAL, 48443; 1 5/8 SH 14G 20'GAL, 48443; 1 5/8 SH 14G 20'GAL, 48443; 1 5/8 SH 14G
20'GAL, 48443; 1 5/8 SH 14G 20'GAL, 48443; 1 5/8 SH 14G 20'GAL, 48443; 1 5/8 SH 14G 20'GAL]
00108700;1/4X4 CF C1018 12FT   [00108700;1/4X4 CF C1018 12FT]
T-Slotted Aluminum Extrusion   [T-Slotted Aluminum Extrusion]
04214849;MC4575M2 Ace Control  [04214849;MC4575M2 Ace Control]
6200604;LOW-CARBON FLAT STOCK;  [6200604;LOW-CARBON FLAT STOCK;, 6200604;LOW-CARBON FLAT STOCK;, 6201602;LOW-CARBON FLAT STOCK;, 6201602;LOW-
CARBON FLAT STOCK;, 6200604;LOW-CARBON FLAT STOCK;, 6201602;LOW-CARBON FLAT STOCK;, 6201602;LOW-CARBON FLAT STOCK;, 6200604;LOW-CARBON FLAT
STOCK;]
00202350;3/4 RD CF C1018 20 FT  [00202350;3/4 RD CF C1018 20 FT]
07005805;5/8 A-36 PLATE 48 X 96 IN     [07005805;5/8 A-36 PLATE 48 X 96 IN]
00108400;1/4 X 3-1/4 CF C1018 12 FT     [00108400;1/4 X 3-1/4 CF C1018 12 FT, 00108400;1/4 X 3-1/4 CF C1018 12 FT, 00107900 ; 1/4 X 2 CF C1018
12 FT, 00108400;1/4 X 3-1/4 CF C1018 12 FT]
"11452-01835;2""X2""X10' ANGLE IRON"    ["11452-01835;2""X2""X10' ANGLE IRON", "11452-01835;2""X2""X10' ANGLE IRON", "11452-01835;2""X2""X10'
ANGLE IRON"]
"11452-02155;1/2"" ODX1/4"" IDX 1.1"    ["11452-02155;1/2"" ODX1/4"" IDX 1.1", "11452-02155;1/2"" ODX1/4"" IDX 1.1", "11452-02155;1/2""
ODX1/4"" IDX 1.1", "11452-02155;1/2"" ODX1/4"" IDX 1.1", "11452-02155;1/2"" ODX1/4"" IDX 1.1", "11452-02155;1/2"" ODX1/4"" IDX 1.1"]
04548820;5 X 5 X 1/2 A-36 ANGLE 20 FT   [04548820;5 X 5 X 1/2 A-36 ANGLE 20 FT]
"11452-01952;5""X3"" X1/4"" 8' ANGLE"   ["11452-01952;5""X3"" X1/4"" 8' ANGLE", "11452-01952;5""X3"" X1/4"" 8' ANGLE", "11452-01952;5""X3""
X1/4"" 8' ANGLE", "11452-01952;5""X3"" X1/4"" 8' ANGLE"]
PTS-NE70-20055;PTS-NE70-20055; magnet   [PTS-NE70-20055;PTS-NE70-20055; magnet]
"12245-00436;2"" X 100' MAGNETIC TAPE"  ["12245-00436;2"" X 100' MAGNETIC TAPE", "12245-00436;2"" X 100' MAGNETIC TAPE", "12245-00436;2"" X
100' MAGNETIC TAPE", "12245-00436;2"" X 100' MAGNETIC TAPE"]
11215-02821;1/2F-3/8M Pin Sq DR Adapter [11215-02821;1/2F-3/8M Pin Sq DR Adapter]
"11452-01623;48"" 13.5-15 VDC EZ TRACK" ["11452-01623;48"" 13.5-15 VDC EZ TRACK"]
11215-02821;1/2F-3/8M Pin Sq DR Adapter [11215-02821;1/2F-3/8M Pin Sq DR Adapter, 11215-02821;1/2F-3/8M Pin Sq DR Adapter]
"11452-01647;11GA 1""OD 8FTL STEEL SQUARE TUBE" ["11452-01649;11GA 1""ODL STEEL X FRAME", "11452-01649;11GA 1""ODL STEEL X FRAME",
"11452-01649;11GA 1""ODL STEEL X FRAME", "11452-01647;11GA 1""OD 8FTL STEEL SQUARE TUBE", "11452-01647;11GA 1""OD 8FTL STEEL SQUARE TUBE",
"11452-01647;11GA 1""OD 8FTL STEEL SQUARE TUBE", "11452-01647;11GA 1""OD 8FTL STEEL SQUARE TUBE", "11452-01649;11GA 1""ODL STEEL X FRAME",
"11452-01647;11GA 1""OD 8FTL STEEL SQUARE TUBE", "11452-01647;11GA 1""OD 8FTL STEEL SQUARE TUBE", "11452-01649;11GA 1""ODL STEEL X FRAME",
"11452-01649;11GA 1""ODL STEEL X FRAME", "11452-01649;11GA 1""ODL STEEL X FRAME", "11452-01649;11GA 1""ODL STEEL X FRAME", "11452-01647;11GA
```

the strings that are matched are written in brackets(values) and randomly one string is taken from those braces and is considered as “key”.

Screenshot-7

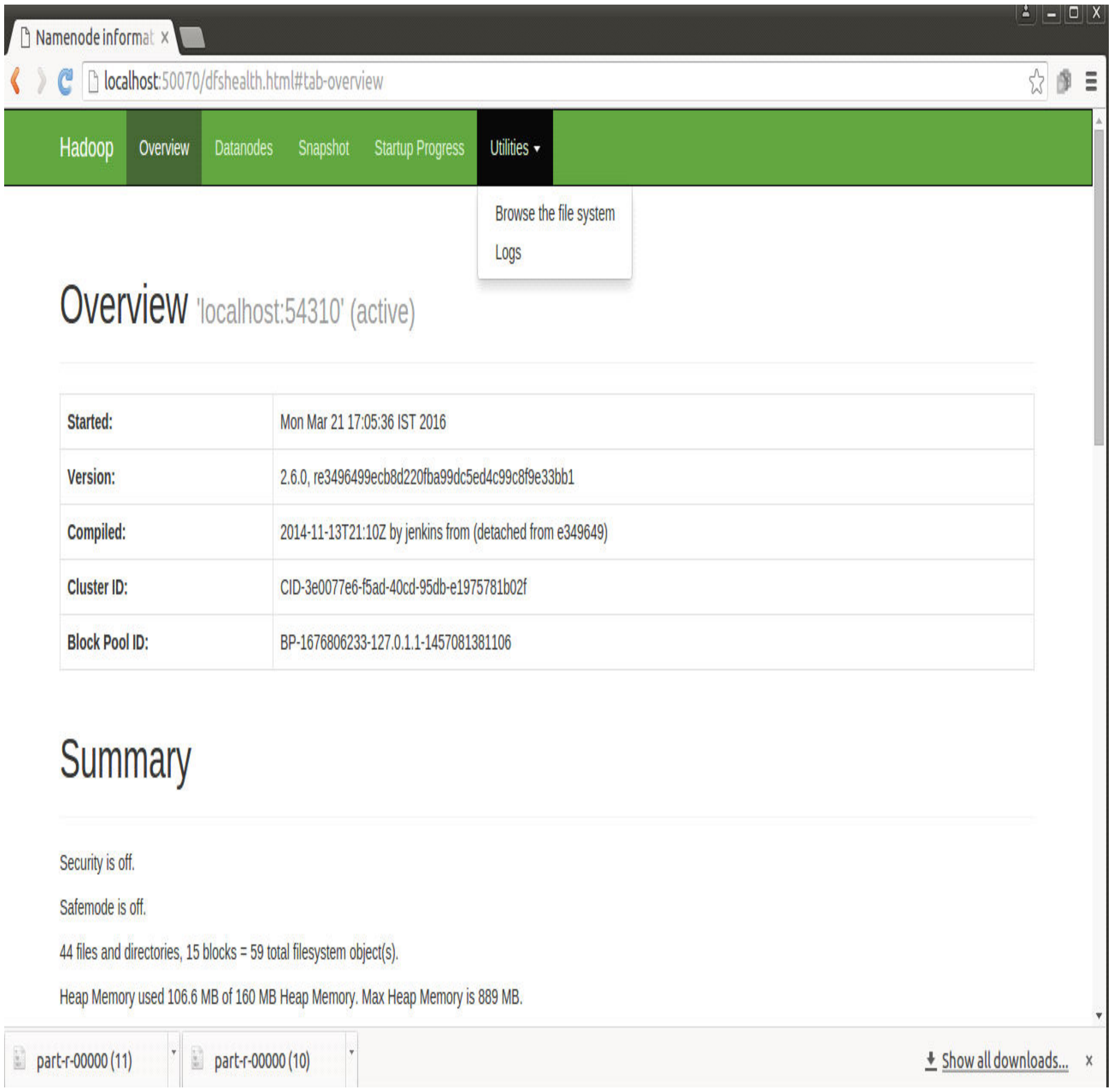


```
1""OD 8FTL STEEL SQUARE TUBE", "11452-01647;11GA 1""OD 8FTL STEEL SQUARE TUBE"]
11215-02821;1/2F-3/8M Pin Sq DR Adapter [11215-02821;1/2F-3/8M Pin Sq DR Adapter, 11215-02821;1/2F-3/8M Pin Sq DR Adapter]
"06000244;06000244, .375 0-1 Drill Rod" ["06000244;06000244, .375 0-1 Drill Rod"]
"11452-01623;48"" 13.5-15 VDC EZ TRACK" ["11452-01623;48"" 13.5-15 VDC EZ TRACK"]
11215-02821;1/2F-3/8M Pin Sq DR Adapter [11215-02821;1/2F-3/8M Pin Sq DR Adapter]
"11452-01623;48"" 13.5-15 VDC EZ TRACK" ["11452-01623;48"" 13.5-15 VDC EZ TRACK", "11452-01623;48"" 13.5-15 VDC EZ TRACK"]
89820252;TEMPORARY STRUCTURE PARTITIONS;       [89820252;TEMPORARY STRUCTURE PARTITIONS;]
12801300;12801300; 1-1/4X21' SCH 40 PIPE       [12801300;12801300; 1-1/4X21' SCH 40 PIPE]
"11452-01945;11GA 2""X20 FT SQUARE TUBE"       ["11452-01945;11GA 2""X20 FT SQUARE TUBE", "11452-01945;11GA 2""X20 FT SQUARE TUBE"]
89820252;TEMPORARY STRUCTURE PARTITIONS;       [89820252;TEMPORARY STRUCTURE PARTITIONS;]
"11452-01644;1/8""T 2"" 2"" STEEL ANGLE"       ["11452-01694;1/8""T 1"" 1"" STEEL ANGLE", "11452-01694;1/8""T 1"" 1"" STEEL ANGLE",
"11452-01694;1/8""T 1"" 1"" STEEL ANGLE", "11452-01644;1/8""T 2"" 2"" STEEL ANGLE", "11452-01644;1/8""T 2"" 2"" STEEL ANGLE",
"11452-01694;1/8""T 1"" 1"" STEEL ANGLE", "11452-01694;1/8""T 1"" 1"" STEEL ANGLE", "11452-01644;1/8""T 2"" 2"" STEEL ANGLE",
"11452-01644;1/8""T 2"" 2"" STEEL ANGLE", "11452-01644;1/8""T 2"" 2"" STEEL ANGLE", "11452-01694;1/8""T 1"" 1"" STEEL ANGLE",
"11452-01694;1/8""T 1"" 1"" STEEL ANGLE", "11452-01694;1/8""T 1"" 1"" STEEL ANGLE", "11452-01644;1/8""T 2"" 2"" STEEL ANGLE",
"11452-01644;1/8""T 2"" 2"" STEEL ANGLE", "11452-01644;1/8""T 2"" 2"" STEEL ANGLE"]
89820252;TEMPORARY STRUCTURE PARTITIONS;       [89820252;TEMPORARY STRUCTURE PARTITIONS;, 89820252;TEMPORARY STRUCTURE PARTITIONS;,
89820252;TEMPORARY STRUCTURE PARTITIONS;]
"11452-01945;11GA 2""X20 FT SQUARE TUBE"       ["11452-01945;11GA 2""X20 FT SQUARE TUBE"]
"18081300;2 X 2 X 1/8 304 SS ANGLE 22 FT       ["18081300;2 X 2 X 1/8 304 SS ANGLE 22 FT]
"11452-01945;11GA 2""X20 FT SQUARE TUBE"       ["11452-01945;11GA 2""X20 FT SQUARE TUBE"]
89820252;TEMPORARY STRUCTURE PARTITIONS;       [89820252;TEMPORARY STRUCTURE PARTITIONS;, 89820252;TEMPORARY STRUCTURE PARTITIONS;]
04500520;3/4 X 3/4 X 1/8 A-36 ANGLE 20 FT      [04500520;3/4 X 3/4 X 1/8 A-36 ANGLE 20 FT]
"PMS-51289;PMS-51289, .500 0-1 Drill Rod"      ["PMS-51289;PMS-51289, .500 0-1 Drill Rod"]
"04505320;2 X 2 X 3/16 A-36 ANGLE 20 FT"               ["04505320;2 X 2 X 3/16 A-36 ANGLE 20 FT"       ]
21429880; 1/4 X 1 6061-T6511 EXT ALUM 12 FT    [21430040;1/4 x 3 6061-T6511 EXT ALUM 12ft, 21429960; 1/4 X 2 6061-T6511 EXT ALUM 12 FT,
21430040;1/4 x 3 6061-T6511 EXT ALUM 12ft, 21430040;1/4 x 3 6061-T6511 EXT ALUM 12ft, 21430080; 1/4 X 4 6061-T6511 EXT ALUM 12 FT, 21429880;
1/4 X 1 6061-T6511 EXT ALUM 12 FT]
06215500;06215500;1/2 RD A-36 HR STEEL 20FT    [06215500;06215500;1/2 RD A-36 HR STEEL 20FT]
0951443;1/4-20 Black Zinc 10S Drop In T-Nut    [0951443;1/4-20 Black Zinc 10S Drop In T-Nut, 0951443;1/4-20 Black Zinc 10S Drop In T-Nut,
0951443;1/4-20 Black Zinc 10S Drop In T-Nut, 0951443;1/4-20 Black Zinc 10S Drop In T-Nut, 0951443;1/4-20 Black Zinc 10S Drop In T-Nut,
0951443;1/4-20 Black Zinc 10S Drop In T-Nut, 0951443;1/4-20 Black Zinc 10S Drop In T-Nut, 0951443;1/4-20 Black Zinc 10S Drop In T-Nut]
MAX30-B200415AIMA-000;MAX30-B200415AIMA-000    [MAX30-B200415AIMA-000;MAX30-B200415AIMA-000]
```

(continued output of 6.3.2)

7.4 HTML PAGE

Screenshot-8



Overview 'localhost:54310' (active)

Started:	Mon Mar 21 17:05:36 IST 2016
Version:	2.6.0, re3496499ecb8d220fba99dc5ed4c99c8f9e33bb1
Compiled:	2014-11-13T21:10Z by jenkins from (detached from e349649)
Cluster ID:	CID-3e0077e6-f5ad-40cd-95db-e1975781b02f
Block Pool ID:	BP-1676806233-127.0.1.1-1457081381106

Summary

Security is off.

Safemode is off.

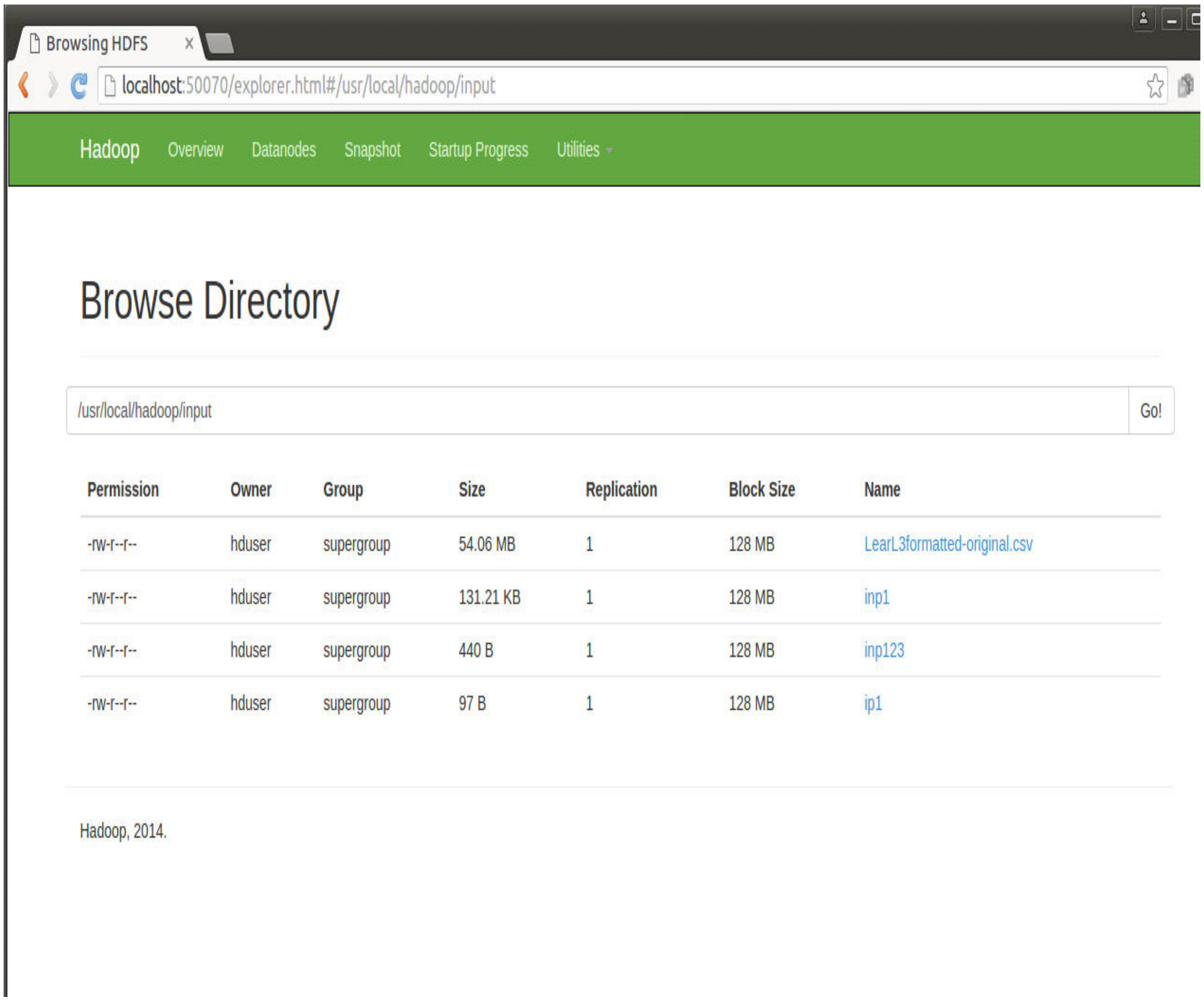
44 files and directories, 15 blocks = 59 total filesystem object(s).

Heap Memory used 106.6 MB of 160 MB Heap Memory. Max Heap Memory is 889 MB.

when we wanted to know the output we need to login in the localhost:50070/dfshealth.html#tab-overview and goto utilities then browse the file system

7.5 BROWSING THE HDFS FOR INPUT

Screenshot-9



Browsing HDFS

localhost:50070/explorer.html#/usr/local/hadoop/input

Browse Directory

/usr/local/hadoop/input

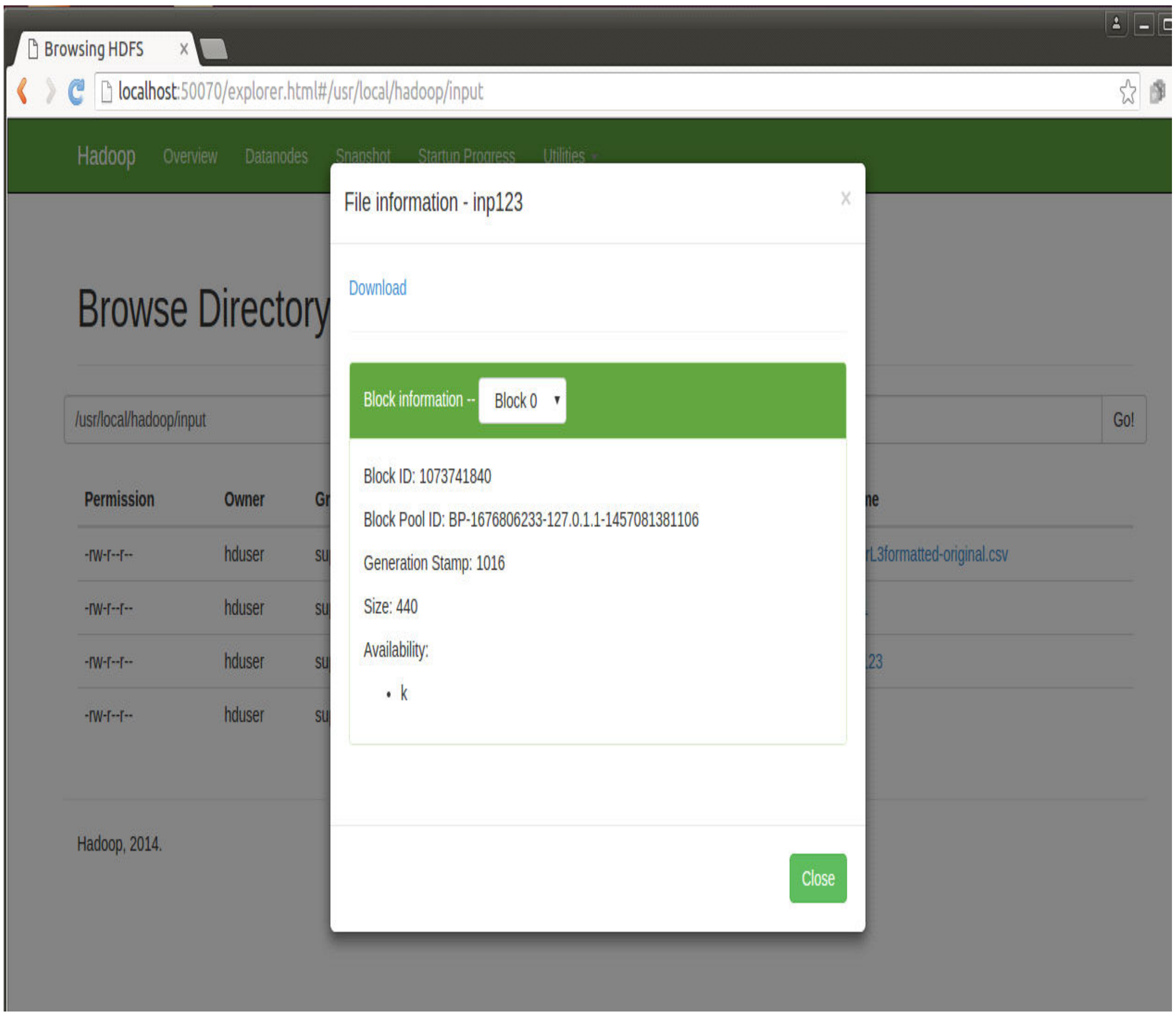
Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	hduser	supergroup	54.06 MB	1	128 MB	LearL3formatted-original.csv
-rw-r--r--	hduser	supergroup	131.21 KB	1	128 MB	inp1
-rw-r--r--	hduser	supergroup	440 B	1	128 MB	inp123
-rw-r--r--	hduser	supergroup	97 B	1	128 MB	ip1

Hadoop, 2014.

(how input is browsed in the directory)

7.6 DOWNLOAD THE OUTPUT

Screenshot-10



File information - inp123

Block information -- Block 0

Block ID: 1073741840

Block Pool ID: BP-1676806233-127.0.1.1-1457081381106

Generation Stamp: 1016

Size: 440

Availability:

- k

Permission	Owner	Gr
-rw-r--r--	hduser	su
-rw-r--r--	hduser	su
-rw-r--r--	hduser	su
-rw-r--r--	hduser	su

(the file information is displayed)

SYSTEM TESTING

6. SYSTEM TESTING

Testing is the process of detecting errors. Testing performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later on during maintenance also.

Testing Objectives include:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say,

1. Testing is a process of executing a program with the intent of finding an error
2. A good test case is one that has a probability of finding an as yet undiscovered error
3. A successful test is one that uncoversan undiscovered error.

TESTING PRINCIPLES:

- All tests should be traceable to end user requirements.
- Tests should be planned long before testing begins.
- Testing should begin on a small scale and progress towards testing in large.
- Exhaustive testing is not possible.
- To be most effective testing should be conducted by a independent third party.

TESTING STRATEGIES

A Strategy for software testing integrates software test cases into a series of well planned steps that result in the successful construction of software. Software testing is a broader topic for what is referred to as Verification and Validation. Verification refers to the set of activities that ensure that the software correctly implements a specific function. Validation refers he set of activities that ensure that the software that has been built is traceable to customer's requirements.

UNIT TESTING:

Unit testing focuses verification effort on the smallest unit of software design that is the module. Using procedural design description as a guide, important control paths are tested to uncover errors within the boundaries of the module. The unit test is normally white box testing oriented and the step can be conducted in parallel for multiple modules.

WHITE BOX TESTING:

This is a unit testing method where a unit will be taken at a time and tested thoroughly at a statement level to find the maximum possible errors. The white box testing is also called Glass Box Testing. This testing is done step wise where every piece of code is tested, taking care that every statement in the code is executed at least once.

BLACK BOX TESTING:

This testing method considers a module as a single unit and checks the unit at interface and communication with other modules rather getting into details at statement level. Here the module will be treated as a black box that will take some input and generate output. Output for a given set of input combinations are forwarded to other modules.

INTEGRATION TESTING:

Integration testing is a systematic technique for constructing the program structure, while conducting test to uncover errors associated with the interface. The objective is to take unit tested methods and build a program structure that has been dictated by design.

VALIDATION TESTING:

At the end of integration testing software is completely assembled as a package. Validation testing is the next stage, which can be defined as successful when the software functions in the manner reasonably expected by the customer. Reasonable expectations are those defined in the software requirements specifications. Information contained in those sections form a basis for validation testing approach.

SYSTEM TESTING:

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all system elements have been properly integrated to perform allocated functions.

CONCLUSION

8. CONCLUSIONS

The result is totally based on comparator class. The input data need not be sorted and input consists of several lines of data depending upon the situation but the output data will be grouped upon similarity. If in future, if we get another ways of grouping in comparator class there can be another ways of grouping results. Right now Hadoop has limited functionalities i.e user has less control over it, may be in future we can have more functionalities.

REFERENCES

REFERENCES

1. Hadoop: The Definitive Guide, 3rd Edition, Tom White, O'Reilly Media, Inc.
2. http://www.tutorialspoint.com/hadoop/
3. https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
4. http://stackoverflow.com/questions/14728480/what-is-the-use-of-grouping-comparator-in-hadoop-map-reduce
5. http://stackoverflow.com/questions/16184745/what-is-difference-between-sort-comparator-and-group-comparator-in-hadoop
6. http://www.levenshtein.net/levenshtein_links.htm
7. https://en.wikipedia.org/wiki/Levenshtein_distance
8. https://www.youtube.com/watch?v=We3YDTzNXEk
9. https://www.youtube.com/watch?v=aEIhvv5p-V8
10. http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/
11. http://www.bogotobogo.com/Hadoop/BigData_hadoop_Install_on_ubuntu_single_node_cluster.php
12. https://www.youtube.com/watch?v=Nb1sinaTlmo
13. https://hadoop.apache.org/docs/r1.2.1/single_node_setup.html
14. http://saphanatutorial.com/hadoop-installation-on-windows-7-using-cygwin/
15. https://www.youtube.com/watch?v=2SU4qZwz8HU
16. http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/indigosr2
17. https://www.youtube.com/watch?v=IBzcRpN9Zo4
18. https://dzone.com/articles/running-hadoop-mapreduce
19. https://letsdobigdata.wordpress.com/2013/12/07/running-hadoop-mapreduce-application-from-eclipse-kepler/
20. https://www.youtube.com/watch?v=hJsaChh2Yhk