Analysis of Youtube Data to Identify the core interests of the population across the world

Deep Roychoudhury
X19109130
MSc in Cloud Computing
National College of Ireland
Dublin, Ireland
x19109130@student.ncirl.ie

***Abstract*— People across various parts of the world have different interests and hobbies. To improve their skills, people take the help of videos from Youtube and many other video streams. The data about the views, likes, comments, etc are collected and saved as datasets. In this paper, these datasets are analysed to identify the interests of people spread across India, the US, Russia, France, and Canada. This analysis might help the government or even businesses to understand the hobbies and create a plan of action. Understanding the categories of videos could also help YouTubers to understand the type of content creation required to make an impact on the market.**

I. INTRODUCTION

The main goal of business is to implement a strategy that will make an impact on the market. To make an impact on the market, companies need to understand the hobbies and interests of people. Similar to business, the government can also help in improvising and providing necessities to people by understanding the requirements of people. During their free time, people usually work on their hobbies and try to improve their understanding and knowledge on that subject by watching youtube videos. Youtube is a worldwide application which is an American online video sharing platform and is available to people free of cost. In this paper, pen source youtube datasets are used from Kaggle for the year 2017-2018. Datasets of 5 different countries have been taken (India, US, Canada, France, and Russia). These datasets will be analysed using map-reduce to provide information about the views based upon different categories ids. Along with views, likes, dislikes and comments will be analysed to provide an analysis of the interaction level of different countries. Finally, median and standard deviation will be identified to check the variation in different categories. These results will help in identifying the interests of people across different parts of the world.

II. RELATED WORK

The data obtained from various sources are stored in databases that are used later for processing or analysing. The storage of data can be done either on relational databases or non-relational databases. The relational databases are MySQL, PostgreSQL, etc. whereas the non-relational databases are hbase(hadoop), Cassandra, mongodb, etc. The structured data are usually stored in relational databases whereas the unstructured data are stored in non-relational databases.

Cassandra is a non-relational database that was originally developed by Facebook and is currently open-source. It promotes horizontal scalability using a column-oriented data model [1]. Cassandra provides high availability, fault tolerance, and high performance. Cassandra uses keyspaces and columns for its storage because of which it is schema-free. Hence, Cassandra would be the best option to use apart from other NoSQL databases.

PostgresSQL is a relational database that can be used to store the analysed data for later retrieval. The latest version of PostgreSQL can store data in JSON format and the entire data will be stored in a single JSON object. It is the most advanced open-source database system [2]. Amazon provides a perfect source for PostgreSQL. Amazon provides AWS RDS as a service which has various kinds of relational database. In this paper, AWS RDS will be used for the relational database.

SQL databases provide tables with fixed rows and columns whereas NoSQL provides rows and dynamic columns. NoSQL is used for large data with simple lookup queries and is widely used for analysing and traversing relationships between connected data. SQL databases provide vertical scaling whereas NoSQL provides horizontal scaling and is much more flexible [4].

MapReduce Framework works on two critical operations: map and reduce. Map function makes a key-value pair of a dataset and sends the key-value pair dataset to the reducer. The reducer then utilizes the key-value pair from the mapper and performs the required operation. The reducer also utilizes tuple class to persist the operational data. The input dataset is usually stored in HDFS (Hadoop Distributed File System) and is then transferred to the mapper class.

Hadoop makes use of YARN which is a cluster resource management. YARN uses Resource Manager to interact with Node Managers which allocates containers for distributed computing. The computing resources have their controllers, the Application Master [3]. Distributed storage is provided by HDFS.

A similar study was conducted for analysing the youtube data and a comparison was made using MapReduce, Pig, and Hive [5]. The data analysed was only limited to a specific country dataset. Hive and Pig are script-based Query Languages. Whereas, MapReduce can be implemented using C, Java, and Python. Based on performance it is found that a well-written MapReduce function works faster than that of Pig and Hive. Hence, MapReduce is the choice in this paper as well since its faster, language-independent, fault-tolerant, and can be parallelized automatically. MapReduce works on key-value pairs. The paper [5] focuses on personal level interpretation for top channels and top videos. The analysis can be carried out to find out the basic requirements and hobbies of people in different countries.

Hadoop has been written in Java. Java is an object-oriented language and the programs written in java are platform-independent. Java provides a platform for distributed computing which involves the usage of several computers or VMs in a network working together. On the other hand, Python is a user-friendly language with high productivity and speed. Due to its ability to integrate with third-party modules, it is best suited for showing the results in graphical formats. Hence, in this paper, Java has been chosen as the MapReduce programming language whereas Python has been chosen for graphical representation.

III. Methodology

A. Datasets

The open-source youtube datasets have been fetched from Kaggle. The data fetched from Kaggle are India, the United States, Canada, France, and Russia. The datasets fetched contains data from 2017-2018. The datasets give information about the video id, views, video category, number of likes, dislikes, comment counts, tags, etc. It consists of a total of 16 columns each and over 40000 rows on each dataset. The category names have been provided by Kaggle with respect to a category id in JSON format. The data will be categorized based upon category id. Analysis of data will be carried based upon view counts, uploads, likes, dislikes, comments, etc.

B. Data Processing Activities

Once the datasets are loaded to the Cassandra database, the processing of the datasets start. There are three types of data processing which takes place in the project.

a) Top 10:

In the first operation, the data is split with category id in the mapper class. The split category id with the country is assigned '1' and is then passed onto reducer. The reducer class reduces the task and gets the total count or upload counts of videos based upon category id and the country name. Finally, the top 10 categories are identified for each country. The output is then stored in PostgreSQL.

b) Views:

In the second operation, similar to top10 approach views are counted based upon category ids of each country on the reducer class. The reducer class is used to find out the median and standard deviation of view counts identified by different countries. The view counts along with standard deviation and median are stored in PostgreSQL. The final output of the standard deviation and median is stored in a separate table to cross-check the output.

c) Interaction:

The third operation is done by counting the total number of likes, dislikes, and comment count. This indicates the interactive nature of people. The total interaction count is the sum of all the three parameters. The output is stored in a PostgreSQL table. The median and standard deviation of the interaction count are found out and stored in a database similar to that of view operation.

Hence, the standard deviation and median will help in identifying the difference between each category and based on the country. The data will also help in identifying the interests of people from different countries..

C. Technologies

a) Java with Maven:

Java is an object-oriented platform-independent language. Java is highly reliable, easy to debug, and has many libraries to support its operations. Programs written in Java are portable and are less likely to crash in comparison to other programming languages[6]. Java provides type safety and garbage collection because of which memory leaks are less. The processing power, efficiency, and storage are much faster in Java when compared to other programming languages. The Java version used in this experiment is Java8. Maven has been used for the automatic generation of JAR files for Hadoop operations. Below is the setup in the maven for the same.

```
<plugin>
    <artifactId>maven-assembly-plugin</artifactId>
    <executions>
            <execution>
                    <phase>package</phase>
                    <goals>
                            <goal>single</goal>
                    </goals>
            </execution>
    </executions>
    <configuration>
            <descriptorRefs>
                 <descriptorRef>jar-with-
dependencies</descriptorRef>
            </descriptorRefs>
    </configuration>
  </plugin>
```

b) Apache Hadoop

Hadoop is originally built on top of Nutch which is a highly extensible and scalable open-source web crawler. Nutch is itself written in Java and is the parent of Hadoop. Hence, it is platform-independent and is modular. The Hadoop dependency used in this project with Java is 1.2.1. Whereas Hadoop 2.9.2 has been used locally which makes use of Yarn to run the code.

```
<dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-core</artifactId>
   <version>1.2.1</version>
</dependency>
```

c) Cassandra

Cassandra is highly scalable and elastic at the same time. It is an open-source platform. This non-relational database follows peer-to-peer connection architecture because of which the failure rate is less. The Cassandra version used in this project is 3.8.0.

```
<dependency>
        <groupId>com.datastax.cassandra</groupId>
        <artifactId>cassandra-driver-core</artifactId>
        <version>3.8.0</version>
</dependency>
```

d) AWS RDS PostgreSQL

PostgreSQL is an open-source relational database that supports ACID(Atomicity, Consistency, Isolation, Durability) properties. PostgreSQL provides many advanced functionalities such as recursive SQL, ordered sets, parallel queries, partitioned tables, etc. AWS provides PostgreSQL as a service of AWS RDS. Also with AWS, the database becomes SSL safe. In this project AWS RDS PostgreSQL has been used for storing the analysed data. The version used in Java project to insert the analysed data to PostgreSQL is 42.2.5.

```
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.2.5</version>
</dependency>
```

e) Python (Matplotlib, numpy, psycopg2)

Python is an open-source object-oriented language. It is dynamic, portable, and provides a high level of productivity. The project uses the Python language to display a graphical representation of MapReduce output. The python version used in this project is 3.6.9. Matplotlib and Numpy have been used to represent the graphical analysis of MapReduce output such as pie chart, scatter plot, stack plot, etc. Psycopg2 has been used as a driver to retrieve data from AWS RDS PostgreSQL.

D. Method of Approach

1) Specification -

The project has been developed using a virtual machine from Oracle VM VirtualBox Manager version 6.1. The operating system installed in the virtual machine is Ubuntu 20.04 64-bit amd processor.
The java installed in the virtual machine is 1.8.0_265. The Eclipse IDE used is of version 2020-06 (4.16.0). The Python version installed is 3.6.9.
Apache Cassandra version 3.8.0 is installed. AWS RDS PostgreSQL was created with the below configuration.
Class - db.t2.micro
Region & AZ - us-east-1f

2) Process –

Youtube datasets from Kaggle have been downloaded for India, United States, Canada, France, and Russia. The datasets are then loaded to the Cassandra database using Cassandra shell. The process to load the datasets to Cassandra is as follows:

1) Starting the Cassandra service – The Cassandra service is started using the below command :

```
sudo service cassandra start
```

2) The Cassandra shell is started using the command *cqlsh*

3) First a KEYSPACE is created with class as SimpleStrategy and replication factor of 1.

```
Create keyspace test with replication =
{'class':'SimpleStrategy','replication_factor': 1};
```

4) Then column families are created with 5 tables for India, United States, Canada, France and Russia. Below is the create query for india.

```
create ColumnFamily youtubeindia (
video_id varchar PRIMARY KEY,
trending_date timestamp,
title varchar,
channel_title varchar,
category_id int,
publish_time timestamp,
tags varchar,
views bigint,
likes bigint,
dislikes bigint,
comment_count bigint,
thumbnail_link varchar,
comments_disabled Boolean,
ratings_disabled Boolean,
video_error_or_removed Boolean,
description varchar
);
```

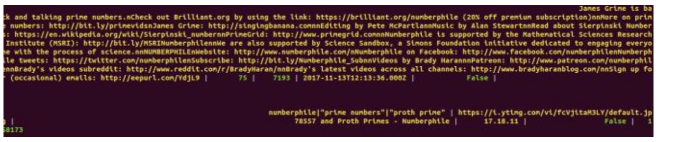
5) The data from csv file is loaded onto the table using the below query. Below is the query for India.

```
copy youtubeindia
(video_id,trending_date,title,channel_title,category
_id,publish_time,tags,views,likes,dislikes,comment
_count,thumbnail_link,comments_disabled,ratings
_disabled,video_error_or_removed,description)
from '/home/hduser/Downloads/INvideos.csv' with
HEADER = TRUE;
```

All these queries are stored in a .cql file (cassandrayoutube.cql). This file is run on the Cassandra shell with the below query.

```
cqlsh –file
/usr/local/cassandra/cassandrayoutube.cql
```

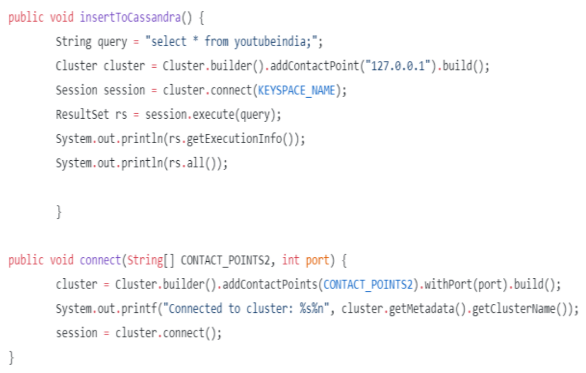
Hence, the data will be loaded onto the Cassandra database.
Below is the table for youtubeindia as seen from the terminal.



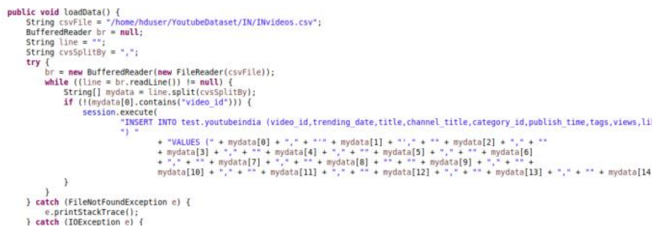
```
                                                                                                          James Grime is ba
ck and talking prime numbers.nCheck out Brilliant.org by using the link: https://brilliant.org/numberphile (20% off premium subscription)nnMore on prim
e numbers: http://bit.ly/primevidsnJames Grime: http://singingbanana.comnnEditing by Pete McPartlannMusic by Alan StewartnnRead about Sierpinski Number
s: https://en.wikipedia.org/wiki/Sierpinski_numbernnPrimeGrid: http://www.primegrid.comnnNumberphile is supported by the Mathematical Sciences Research
 Institute (MSRI): http://bit.ly/MSRINumberphilennWe are also supported by Science Sandbox, a Simons Foundation initiative dedicated to engaging everyo
ne with the process of science.nnNUMBERPHILEnWebsite: http://www.numberphile.com/nNumberphile on Facebook: http://www.facebook.com/numberphilenNumberph
ile tweets: https://twitter.com/numberphilenSubscribe: http://bit.ly/Numberphile_SubnnVideos by Brady HarannnPatreon: http://www.patreon.com/numberphil
ennBrady's videos subreddit: http://www.reddit.com/r/BradyHaran/nnBrady's latest videos across all channels: http://www.bradyharanblog.com/nnSign up fo
r (occasional) emails: http://eepurl.com/YdjL9 |       75 |    7193 | 2017-11-13T12:13:36.000Z |            False |



                                                   numberphile|"prime numbers"|"proth prime" | https://i.ytimg.com/vi/fcVjitaM3LY/default.jp
g |                                                     78557 and Proth Primes - Numberphile |     17.18.11 |                       False |   1
58173
```

Figure 1 – Output of Cassandra select Query

In order to fetch data from the Cassandra database, the below java code has been used.



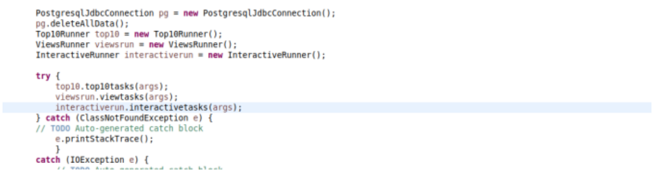
```java
public void insertToCassandra() {
        String query = "select * from youtubeindia;";
        Cluster cluster = Cluster.builder().addContactPoint("127.0.0.1").build();
        Session session = cluster.connect(KEYSPACE_NAME);
        ResultSet rs = session.execute(query);
        System.out.println(rs.getExecutionInfo());
        System.out.println(rs.all());

        }

public void connect(String[] CONTACT_POINTS2, int port) {
        cluster = Cluster.builder().addContactPoints(CONTACT_POINTS2).withPort(port).build();
        System.out.printf("Connected to cluster: %s%n", cluster.getMetadata().getClusterName());
        session = cluster.connect();
}
```



```java
public void loadData() {
    String csvFile = "/home/hduser/YoutubeDataset/IN/INvideos.csv";
    BufferedReader br = null;
    String line = "";
    String cvsSplitBy = ",";
    try {
        br = new BufferedReader(new FileReader(csvFile));
        while ((line = br.readLine()) != null) {
            String[] mydata = line.split(cvsSplitBy);
            if (!(mydata[0].contains("video_id"))) {
                session.execute(
                        "INSERT INTO test.youtubeindia (video_id,trending_date,title,channel_title,category_id,publish_time,tags,views,li
                        ") "
                                + "VALUES (" + mydata[0] + "," + "'" + mydata[1] + "'," + "" + mydata[2] + "," + ""
                                + mydata[3] + "," + "" + mydata[4] + "," + "" + mydata[5] + "," + "" + mydata[6]
                                + "," + "" + mydata[7] + "," + "" + mydata[8] + "" + "" + mydata[9] + "," + "" +
                                mydata[10] + "," + "" + mydata[11] + "," + "" + mydata[12] + "," + "" + mydata[13] + "," + "" + mydata[14
            }
        }
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
```

Figure 2 Cassandra code for insertion

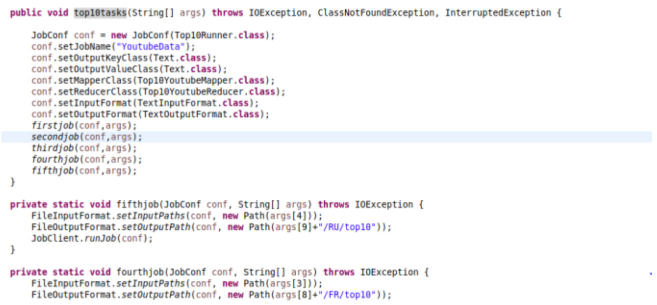
After saving the file in the Cassandra Database, the program deletes the RDS Postgre database, fetches the records, and performs three operations back to back as shown below for top10, views, and interactive.



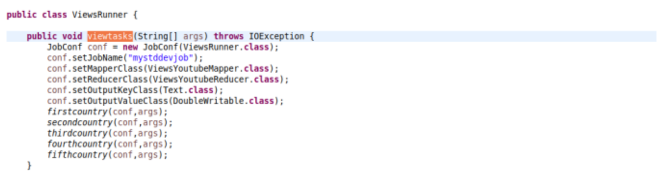
```java
PostgresqlJdbcConnection pg = new PostgresqlJdbcConnection();
pg.deleteAllData();
Top10Runner top10 = new Top10Runner();
ViewsRunner viewsrun = new ViewsRunner();
InteractiveRunner interactiverun = new InteractiveRunner();

try {
    top10.top10tasks(args);
    viewsrun.viewtasks(args);
    interactiverun.interactivetasks(args);
} catch (ClassNotFoundException e) {
// TODO Auto-generated catch block
    e.printStackTrace();
    }
catch (IOException e) {
```

Figure 3 Three Operations for MapReduce

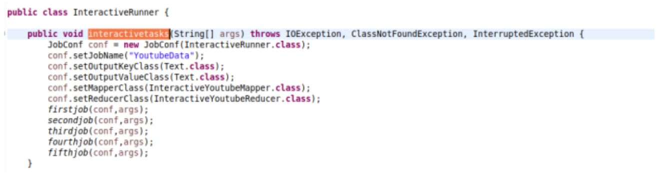
The data is then transferred to the runner classes of top10, interactive and views.



```java
public void top10tasks(String[] args) throws IOException, ClassNotFoundException, InterruptedException {

    JobConf conf = new JobConf(Top10Runner.class);
    conf.setJobName("YoutubeData");
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(Text.class);
    conf.setMapperClass(Top10YoutubeMapper.class);
    conf.setReducerClass(Top10YoutubeReducer.class);
    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);
    firstjob(conf,args);
    secondjob(conf,args);
    thirdjob(conf,args);
    fourthjob(conf,args);
    fifthjob(conf,args);
}

private static void fifthjob(JobConf conf, String[] args) throws IOException {
    FileInputFormat.setInputPaths(conf, new Path(args[4]));
    FileOutputFormat.setOutputPath(conf, new Path(args[9]+"/RU/top10"));
    JobClient.runJob(conf);
}

private static void fourthjob(JobConf conf, String[] args) throws IOException {
    FileInputFormat.setInputPaths(conf, new Path(args[3]));
    FileOutputFormat.setOutputPath(conf, new Path(args[8]+"/FR/top10"));
```



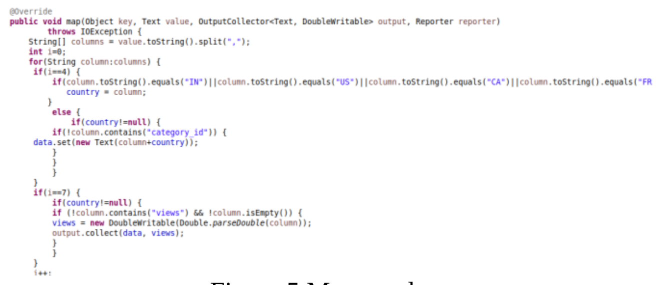
```java
public class ViewsRunner {

    public void viewtasks(String[] args) throws IOException {
        JobConf conf = new JobConf(ViewsRunner.class);
        conf.setJobName("mystddevjob");
        conf.setMapperClass(ViewsYoutubeMapper.class);
        conf.setReducerClass(ViewsYoutubeReducer.class);
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(DoubleWritable.class);
        firstcountry(conf,args);
        secondcountry(conf,args);
        thirdcountry(conf,args);
        fourthcountry(conf,args);
        fifthcountry(conf,args);
    }
```



```java
public class InteractiveRunner {

    public void interactivetasks(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
        JobConf conf = new JobConf(InteractiveRunner.class);
        conf.setJobName("YoutubeData");
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(Text.class);
        conf.setMapperClass(InteractiveYoutubeMapper.class);
        conf.setReducerClass(InteractiveYoutubeReducer.class);
        firstjob(conf,args);
        secondjob(conf,args);
        thirdjob(conf,args);
        fourthjob(conf,args);
        fifthjob(conf,args);
    }
```

Figure 4 Top 10, Views and Interactive Runner class

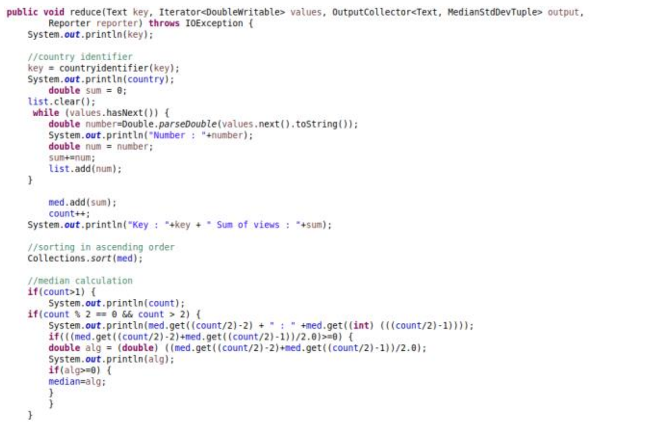
The data from the runner class is then passed onto the mapper class which maps the data in key value pairs.



```java
@Override
public void map(Object key, Text value, OutputCollector<Text, DoubleWritable> output, Reporter reporter)
        throws IOException {
    String[] columns = value.toString().split(",");
    int i=0;
    for(String column:columns) {
     if(i==4) {
        if(column.toString().equals("IN")||column.toString().equals("US")||column.toString().equals("CA")||column.toString().equals("FR
           country = column;
        }
        else {
            if(country!=null) {
        if(!column.contains("category_id")) {
     data.set(new Text(column+country));
        }
        }
        }
     }
     if(i==7) {
        if(country!=null) {
        if (!column.contains("views") && !column.isEmpty()) {
        views = new DoubleWritable(Double.parseDouble(column));
        output.collect(data, views);
        }
        }
     }
     i++;
```

Figure 5 Mapper class

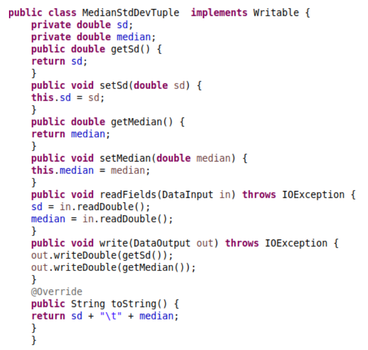
All the three mapper classes are run and the output of the mapper class is then transferred to the reducer class. The reducer class performs the tasks based upon the operations and differentiates each data using the country as shown below.



```java
public void reduce(Text key, Iterator<DoubleWritable> values, OutputCollector<Text, MedianStdDevTuple> output,
        Reporter reporter) throws IOException {
    System.out.println(key);

    //country identifier
    key = countryidentifier(key);
    System.out.println(country);
        double sum = 0;
    list.clear();
     while (values.hasNext()) {
        double number=Double.parseDouble(values.next().toString());
        System.out.println("Number : "+number);
        double num = number;
        sum+=num;
        list.add(num);
    }

        med.add(sum);
        count++;
    System.out.println("Key : "+key + " Sum of views : "+sum);

    //sorting in ascending order
    Collections.sort(med);

    //median calculation
    if(count>1) {
        System.out.println(count);
    if(count % 2 == 0 && count > 2) {
        System.out.println(med.get((count/2)-2) + " : " +med.get((int) (((count/2)-1))));
        if(((med.get((count/2)-2)+med.get((count/2)-1))/2.0)>=0) {
        double alg = (double) ((med.get((count/2)-2)+med.get((count/2)-1))/2.0);
        System.out.println(alg);
        if(alg>=0) {
        median=alg;
        }
        }
    }
```

Figure 6 Reducer Class

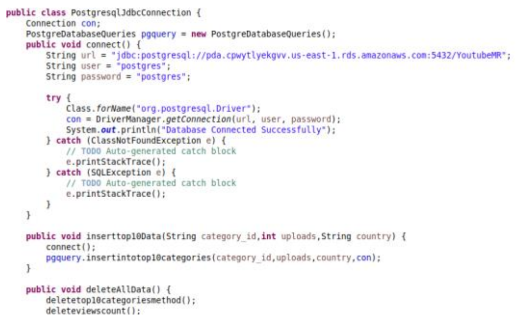
The tuple class MedianStdDevTuple acts as a persistent class to hold the data.



```java
public class MedianStdDevTuple  implements Writable {
    private double sd;
    private double median;
    public double getSd() {
    return sd;
    }
    public void setSd(double sd) {
    this.sd = sd;
    }
    public double getMedian() {
    return median;
    }
    public void setMedian(double median) {
    this.median = median;
    }
    public void readFields(DataInput in) throws IOException {
    sd = in.readDouble();
    median = in.readDouble();
    }
    public void write(DataOutput out) throws IOException {
    out.writeDouble(getSd());
    out.writeDouble(getMedian());
    }
    @Override
    public String toString() {
    return sd + "\t" + median;
    }
    }
```

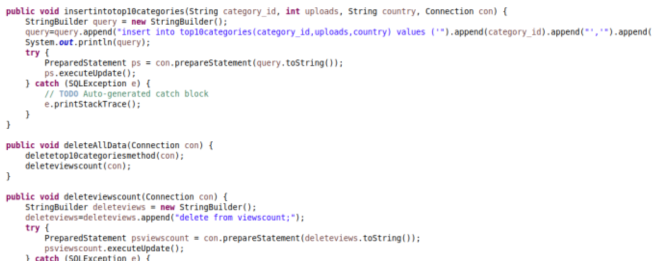
Figure 7 Tuple Class

Hence, the data is saved from the reducer class to the PostgreSQL using first the Jdbc class for Database connection and the Query class to make the query to the database as shown below.



```
public class PostgresqlJdbcConnection {
    Connection con;
    PostgreDatabaseQueries pgquery = new PostgreDatabaseQueries();
    public void connect() {
        String url = "jdbc:postgresql://pda.cpwytlyekgvv.us-east-1.rds.amazonaws.com:5432/YoutubeMR";
        String user = "postgres";
        String password = "postgres";

        try {
            Class.forName("org.postgresql.Driver");
            con = DriverManager.getConnection(url, user, password);
            System.out.println("Database Connected Successfully");
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public void inserttop10Data(String category_id,int uploads,String country) {
        connect();
        pgquery.insertintotop10categories(category_id,uploads,country,con);
    }

    public void deleteAllData() {
        deletetop10categoriesmethod();
        deleteviewscount();
```

Figure 8 Jdbc Connection to Postgre

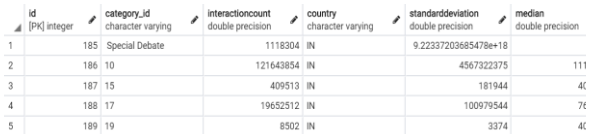


```
public void insertintotop10categories(String category_id, int uploads, String country, Connection con) {
    StringBuilder query = new StringBuilder();
    query=query.append("insert into top10categories(category_id,uploads,country) values ('").append(category_id).append("','").append(
    System.out.println(query);
    try {
        PreparedStatement ps = con.prepareStatement(query.toString());
        ps.executeUpdate();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

public void deleteAllData(Connection con) {
    deletetop10categoriesmethod(con);
    deleteviewscount(con);
}

public void deleteviewscount(Connection con) {
    StringBuilder deleteviews = new StringBuilder();
    deleteviews=deleteviews.append("delete from viewscount;");
    try {
        PreparedStatement psviewscount = con.prepareStatement(deleteviews.toString());
        psviewscount.executeUpdate();
    } catch (SQLException e) {
```

Figure 9 Querying to Postgre

Hence, the data are saved on the AWS RDS. There are total four tables created in Postgre.

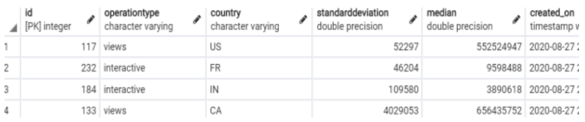
1) Interactioncount – To insert data with the sum of likes,dislikes and comments along with standard deviation and median.



	id [PK] integer	category_id character varying	interactioncount double precision	country character varying	standarddeviation double precision	median double precision
1	185	Special Debate	1118304	IN	9.22337203685478e+18	
2	186	10	121643854	IN	4567322375	111
3	187	15	409513	IN	181944	40
4	188	17	19652512	IN	100979544	76
5	189	19	8502	IN	3374	40

Figure 10 – Interaction PostgreSQL

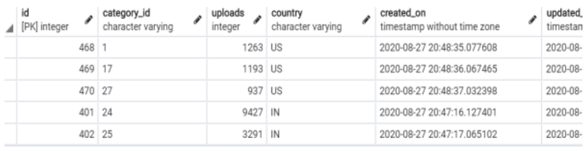
2) Stddevmedian – To cross check the standard deviation and median of views and interactive table. The differentiation on this table can be made using country.



	id [PK] integer	operationtype character varying	country character varying	standarddeviation double precision	median double precision	created_on timestamp v
1	117	views	US	52297	552524947	2020-08-27 2
2	232	interactive	FR	46204	9598488	2020-08-27 2
3	184	interactive	IN	109580	3890618	2020-08-27 2
4	133	views	CA	4029053	656435752	2020-08-27 2

Figure 11 – Standard Deviation and Median PostgreSQL

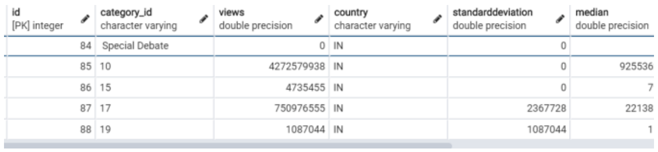
3) Top10categories – To insert the top 10 categories of each country



id [PK] integer	category_id character varying	uploads integer	country character varying	created_on timestamp without time zone	updated_ timestan
468	1	1263	US	2020-08-27 20:48:35.077608	2020-08-
469	17	1193	US	2020-08-27 20:48:36.067465	2020-08-
470	27	937	US	2020-08-27 20:48:37.032398	2020-08-
401	24	9427	IN	2020-08-27 20:47:16.127401	2020-08-
402	25	3291	IN	2020-08-27 20:47:17.065102	2020-08-

Figure 12 – Top 10 categories table PostgreSQL

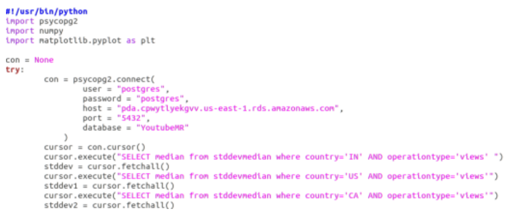
4) Viewscount – To insert the total number of views of each category based upon different countries.



id [PK] integer	category_id character varying	views double precision	country character varying	standarddeviation double precision	median double precision
84	Special Debate	0	IN	0	
85	10	4272579938	IN	0	925536
86	15	4735455	IN	0	7
87	17	750976555	IN	2367728	22138
88	19	1087044	IN	1087044	1

Figure 13 – Views table PostgreSQL

The data after MapReduce are stored in Postgresql which are then fetched using python code to display graphical representation of the output.



```
#!/usr/bin/python
import psycopg2
import numpy
import matplotlib.pyplot as plt

con = None
try:
        con = psycopg2.connect(
                user = "postgres",
                password = "postgres",
                host = "pda.cpwytlyekgvv.us-east-1.rds.amazonaws.com",
                port = "5432",
                database = "YoutubeMR"
            )
        cursor = con.cursor()
        cursor.execute("SELECT median from stddevmedian where country='IN' AND operationtype='views' ")
        stddev = cursor.fetchall()
        cursor.execute("SELECT median from stddevmedian where country='US' AND operationtype='views'")
        stddev1 = cursor.fetchall()
        cursor.execute("SELECT median from stddevmedian where country='CA' AND operationtype='views'")
        stddev2 = cursor.fetchall()
```

Figure 14 – Python code for Graphics

3) *Design Pattern :*

a) External Source Input :

The external source input design pattern has been used so that the data can be directly pulled from an external database (i.e. Cassandra NoSQL) instead of using HDFS as a storage platform. The reason for choosing external source input is that it is much faster to load data to an external database rather than staging the files to HDFS. In this pattern, the InputFormat creates splits which are the Inputsplit objects. The RecordReader sends the InputSplits information using job configuration to Reducer class [7].

b) External Source Output :

The external source output design pattern has been used to write the data outside the HDFS. The outside source chosen for writing the data out are PostgreSQL as well as output Folder(the data is written locally inside folders). This design pattern lets the analysed data to skip the staging of data to HDFS. Hence, it is much faster. The driver class sends the data to mapper class where the splitting of input data to key/value pairs take place. The key/value pair is sent to the reducer class for further manipulation. The key/value pair output is sent directly to the PostgreSQL to be saved. The outputs are also stored in Redis Hash Map instances which is an open-source, key/value store.

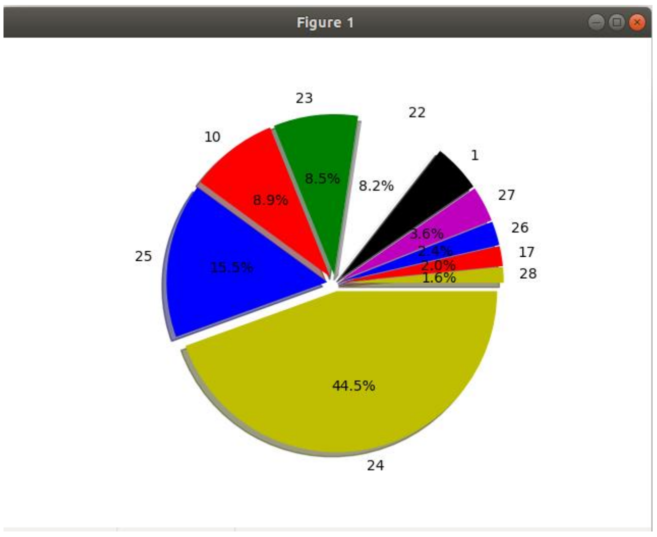
c) Numerical Summarization Pattern :

Numerical Summarization Pattern has been used to hold the values of median and standard deviation in the reducer phase. In this project MedianStdDevTuple has been used where the numerical summarization pattern has been applied to hold the values. The tuple class is a writable object that stores two values median and standard deviation. The reducer calculates the median and standard deviation and stores the final value to Stddevmedian table.

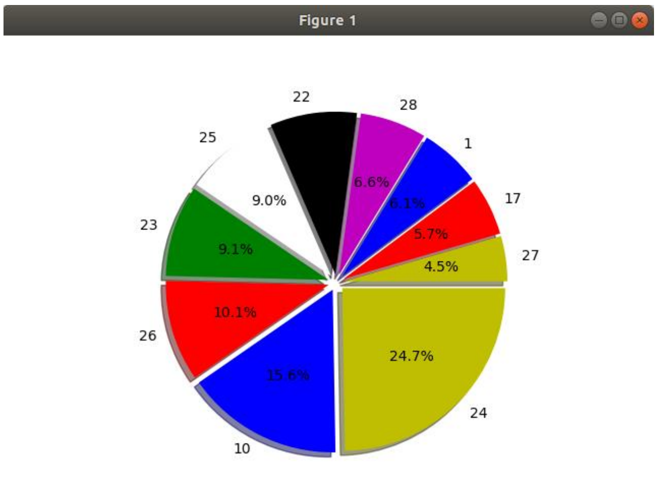
IV. RESULTS

The data processed from MapReduce using Java is fetched using psycopg2 in Python. Matplotlib and NumPy have been used to make plots based on different factors.

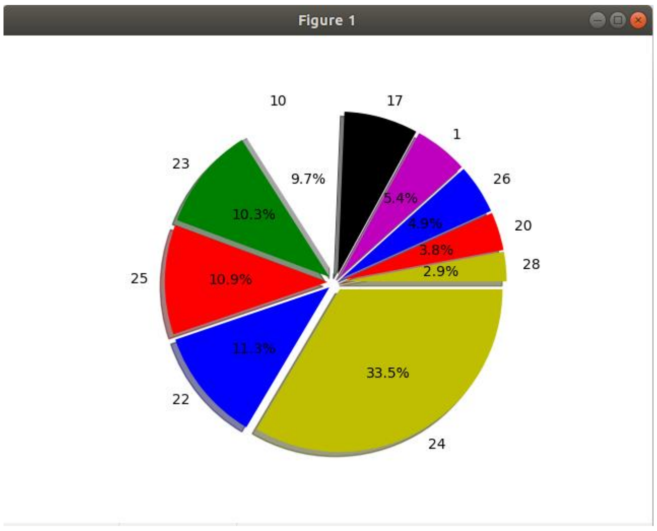
The first result displayed is of the top 10 categories and the percentage of weightage each category id has. Below are the pie charts which show the weightage of each category based upon countries.



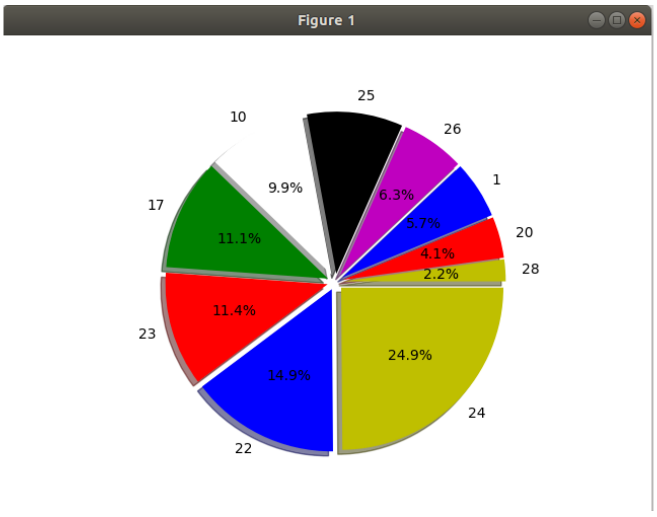
India



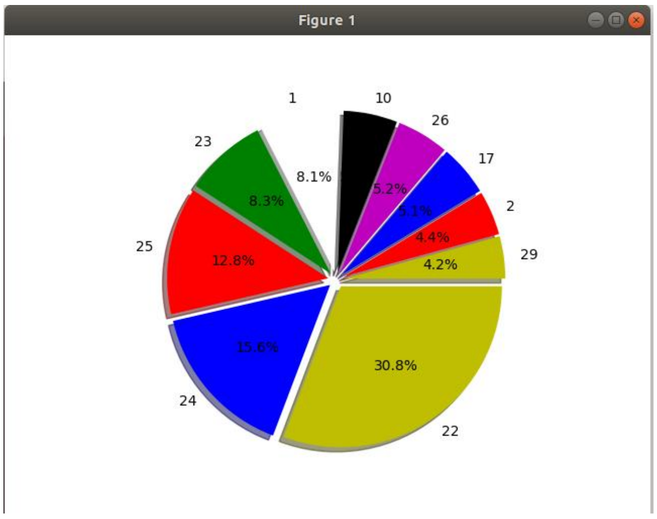
US



Canada



France



Russia

Figure 15 Pie chart of All countries for Top 10

From the pie charts, it can be seen that each category has a different percentage of 5 different countries. For example, category 22 of Russia has 30.8% whereas India has only

8.2% of upload on the same category. It clearly shows the difference in interests of uploading videos.
Below is the sec plot of all the 5 countries. The difference in the upload of the top 10 categories of videos can be clearly seen from the diagram. For example, category 23 has an almost equal number of uploads for France and Canada whereas it is in minority in the US and India.

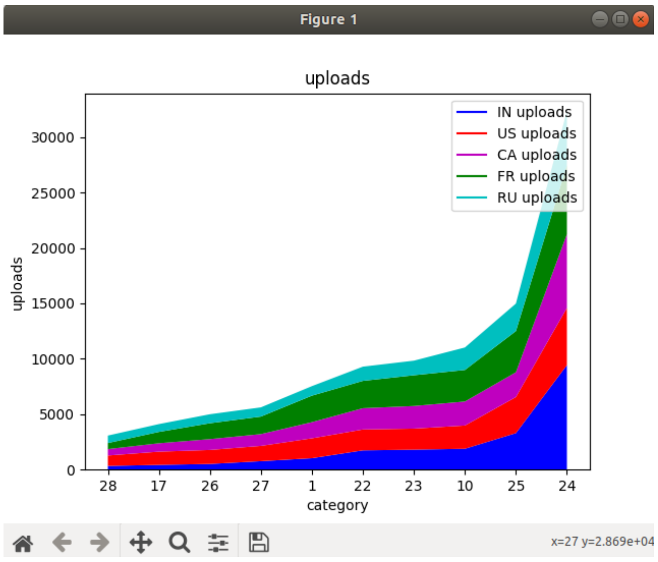


Figure 16 Sec Plot for Top 10 categories

Below is the bar plot of views compared to the median for India. For example below is the scatter plot of India. The scatter plot shows the deviation of various categories from the median value. For Example, the difference in the median value of category 22 and 17 are similar whereas 27 and 20 have a negligible difference.

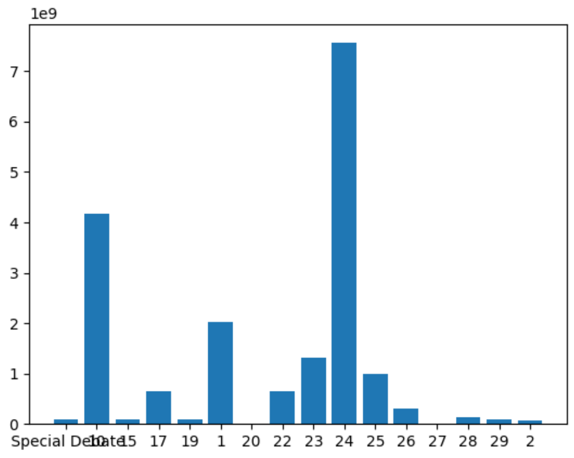


Figure 17 IN

Similar to the median in view count below is the bar plot of the median in interaction count. It can be seen from the figure that the difference in interaction level to that of the median is higher when compared to views count. It simply means that the population in India likes to express their thoughts to only a limited number of categories. Whereas they are more likely to view videos of certain categories rather than making an interaction. For Example, it can be seen that category 27 had the least difference in views whereas it has more difference in interaction.

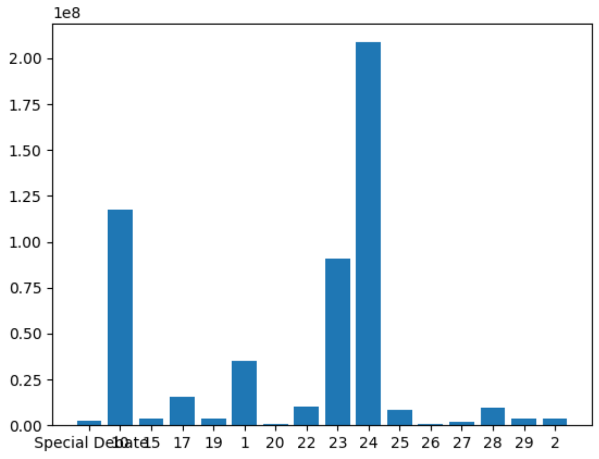


Figure 18

Below is the bar plot for India, for the standard deviation of interaction count. It can be seen that category 24 has the highest difference in the deviation. Whereas, the Special debate category has the least deviation.

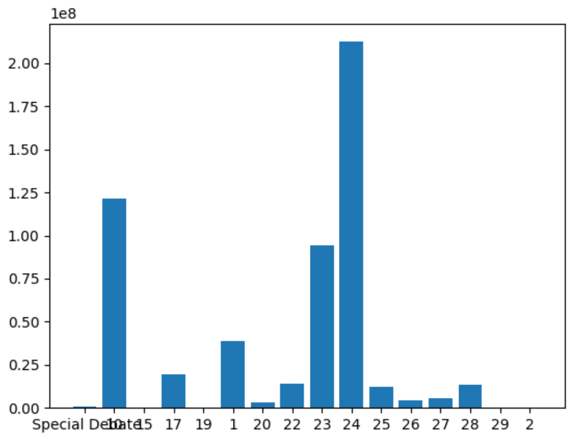


Figure 19 Bar plot w.r.t Standard Deviation interaction count IN

Similar to interaction, bar graph of views with respect to standard deviation can be seen in the below figure.

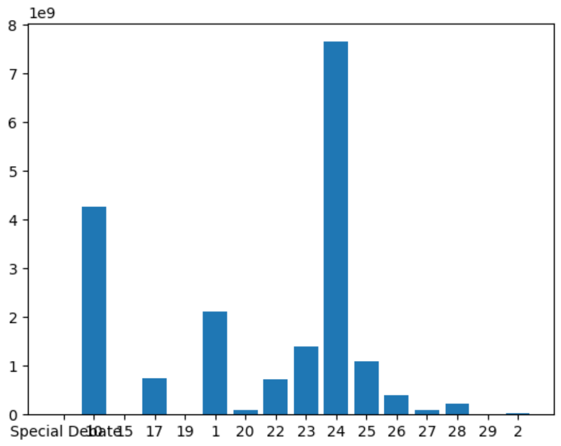
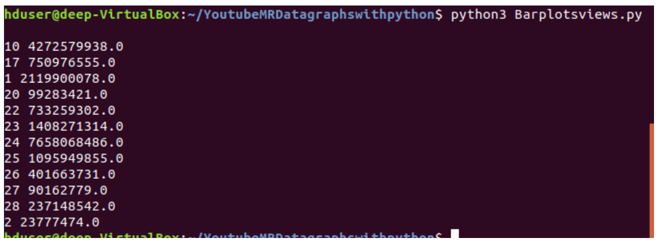


Figure 20 Bar plot w.r.t Standar Deviation views count IN

Comparing both views count and interaction count bar graph it can be seen that the difference of category id 24 is the highest for both views and interaction.
Below is the output of values for the views count standard deviation of India for category ids. Likewise, bar plots can be done for other countries as well to identify the categories people like to interact more and the categories people like to view more.



```
hduser@deep-VirtualBox:~/YoutubeMRDatagraphswithpython$ python3 Barplotsviews.py

10 4272579938.0
17 750976555.0
1 2119900078.0
20 99283421.0
22 733259302.0
23 1408271314.0
24 7658068486.0
25 1095949855.0
26 401663731.0
27 90162779.0
28 237148542.0
2 23777474.0
hduser@deep-VirtualBox:~/YoutubeMRDatagraphswithpython$
```

Figure 21

Similarly, below diagram shows the category id with respect to interaction count of 5 different countries. The difference can be clearly seen from the sec plot.

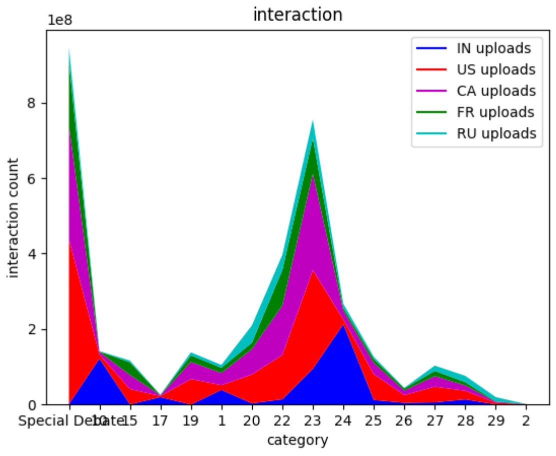


Figure 22 Sec Plot of Interaction Count (5 different countries)

The above sec plot gives us an idea that the majority of US and Canada population interact with the videos of category 23, the majority of video interaction in India is category 24

The sec plot of the view count can be seen in the below figure. The figure shows the difference in view count across different countries with different category id.

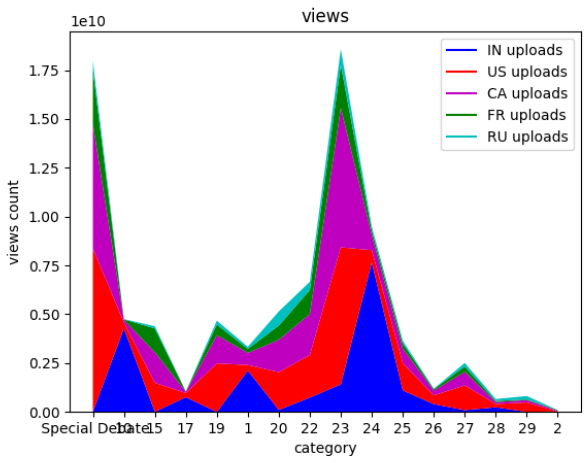


Figure 23

It can be seen that the videos watched at category 23 is higher than interaction for US and Canada.

V. Conclusion and Future Work

The experiment done using Java MapReduce gives an idea about the interests and hobbies of people living in India, the United States, Canada, France, and Russia. Further using median and standard deviation, it is established that some of the video categories get more responses from the people and some don't. The more the response, the more it means people are interested in the videos. Government, YouTubers, and business professionals can utilize this methodology to identify the core interests of the people in their place and take action based on the analysis. Future work in this area will be to introduce a column of subscribers and analyse the data of comments to identify the positive and negative responses. This analysis will help in identifying a more accurate analysis.

VI. Repositories

The MapReduce code repository has been saved to the Github.
Link - https://github.com/DeepRoychoudhury/MRwithYoutubeData

References

[1] E. Dede, B. Sendir, P. Kuzlu, J. Weachock, M. Govindaraju, and L. Ramakrishnan, "Processing Cassandra Datasets with Hadoop-Streaming Based Approaches", IEEE TRANSACTIONS ON SERVICES COMPUTING, VOL. 9, NO. 1, 2016

[2] Shelly Sachdevaa*, Pranav Sharmab, Rupal Jainc, Siddharth Parashar, "Critical Analysis of Data Storage Approaches", ICITETM2020, vol 173, 2020, pp. 264–271.

[3] Thomas C. Bressoud, Qiuyi (Jessica) Tang, “Analysis, Modeling, and Simulation of Hadoop YARN MapReduce”, 2016, IEEE 22nd International Conference on Parallel and Distributed Systems

[4] https://www.mongodb.com/nosql-explained/nosql-vs-sql, MongoDB.

[5] Farzana Shaikh, Danish Pawaskar, Umar Khan, Abutalib Siddiqui, “YouTube Data Analysis using MapReduce on Hadoop”, 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT-2018)

[6] https://www.linkedin.com/pulse/why-hadoop-written-java-astha-srivastava/, linkedin

[7] Donal Miner & Adam Shook, “MapReduce Design Patterns”, O’Reilly