Tipping behavior analysis with Hadoop

Lingyan Zhou Jianhong Zhu

November 10, 2015

Abstract

Taxis are a reliable data source of public transportation and people's tipping behavior. However, the huge amount of data generated by every taxi poses a problem to effectively discover hidden facts from it using conventional computing techniques. Thus we implemented several command-line applications using the Hadoop MapReduce. We incorporated common MapReduce patterns and some special computational techniques that can yield results in one MapReduce job. Our implementations can process five Giga-bytes of compressed taxi trip data in two minutes on average.

Contents

1 Introduction

With the recent advancement in big data technologies, more and more industries now base their businesses and profits on data-driven decision making. For metropolis like New York city, the data generated from its primary mode of transportation, taxi, is a valuable indicator for urban planning, public transportation scheduling and people's daily commuting arrangement. More specifically, taxi companies can analyze the data and direct drivers to more profitable customers, time and areas in greatest demand for fast transit to compete with fast growing star-up companies like Uber and Lyft.

In an attempt to improve the accessibility, transparency, and accountability, the New York Taxi and Limousine Commission has released millions of taxi trip records from 2011[1]. The whole data set contains 540 million records per year. Many researchers have used the taxi trip data set for various purposes. To help others build an intuitive understanding of huge data set, the Chris Whong built a web-based visualization tool to animate every taxi trip on a map[2]. Whong's application inspired developers at ImageWork, they associated flight trips with taxi trips from two major airports and also built a similar visualization tool[3]. Similar applications include [4], a multi-resolution traffic density viewer. Apart from data visualization, there were also attempts to analyze the data statistically. Ryan Hafen did an initial analysis using R using a sample set, which showed basic statistics of trip distances, tip amount and etc[5]. Hafen's work also revealed the relationship between pickup counts and locations[5]. This dataset also attracted academic researches. In [6], Andrew James discussed in details about sanitizing the data set, made comprehensive plots and proposed a ride sharing framework based on pixelized taxi density map.

In this article, we aimed to discover some interesting tipping patterns by examining the June to October, 2013 New York taxi trip data set. In this data set, there are 54 million trip records. Each trip has the following fields: medallion number, hack license number, pickup date and time, drop-off date and time, passenger count, trip duration (in seconds), trip distance, pickup latitude and longitude, drop-off latitude and longitude, tip and fare amount and etc. However, the fare and trip information of every record

are separated in two different files.A record description was distributed along with the data[7].

To effectively compute the statistics, we used Hadoop. Hadoop is a Java implementation of Google's MapReduce framework, which simplifies distributed computing by separating computation in mapping and reducing phases. However, the framework also requires special computational techniques to achieve its maximum performance. For example, each mapper only receives a part of a large data set and there is no way to pass messages between mappers. In this article, we present our attempt to use Hadoop to process and analyze the taxi trip data. We implemented efficient algorithms to make calculation of standard deviations, auto-correlations and etc. finish in one MapReduce job.

2 Methods

In this section, we first preprocessed the data to make it more Hadop-friendly. Then we took a 1% sample of the original data and visualized it using simple matrix plot. This step helped us gain more insight to the data. Next, we detected outliers and erroneous data based on [6]. We implemented several Java command-line application on Hadoop to do the following calculation[8]. In order to determine the quality of the data set, we calculated summary statistics of most fields without removing outliers. Then, with outliers filtered out, we uncovered relationships and dependencies between fields. With the temporal and spacial information in the data set, we found the seasonality of tip amount using autocorrelation, pixelized the average tip amount in regions. Most frequent tip amount and tip ration was also discovered by data binning. Finally, we built a K-nearest-neighbor tip predictor.

2.1 Data Preprocessing

The simplification of MapReduce framework brought everyday programmers to distributed computing. However, the simplified computation model also restricted its application.

One disadvantage of it is to perform join operations of two files. Note that the data we got was separated in two files that needs joining. We wrote a Python script to combine the files line by line and then produce a single compressed file. For easy uploading the data onto the Hadoop clusters, we separated the file into eighteen 200MB chunks.

Note that according to [9], Hadoop splits files into blocks and when running MapReduce jobs, each block will spawn a mapper task. Because the start-up of a mapper takes time and there are also management overhead with each mapper, [9] suggests that the file size should not be too small compared with the block size. In our setup, we used 128MB as the block size. So each 200MB chunk will need 2 mappers, with slightly unbalanced load.

We did not perform data cleansing in this phase. Filtering obvious erroneous records was done on the MapReduce framework to demonstrate the use of filtering pattern (see Section 2.3).

2.2 Visualizing Sample Data

In order to gain more insight to the data, we worked on a smaller subset of the data, with 0.1% sample size. The sample was selected at random without replacement. Like in [5], we performed initial statistical and visual investigation with R. Part of the statistical summary is shown in Table 1.

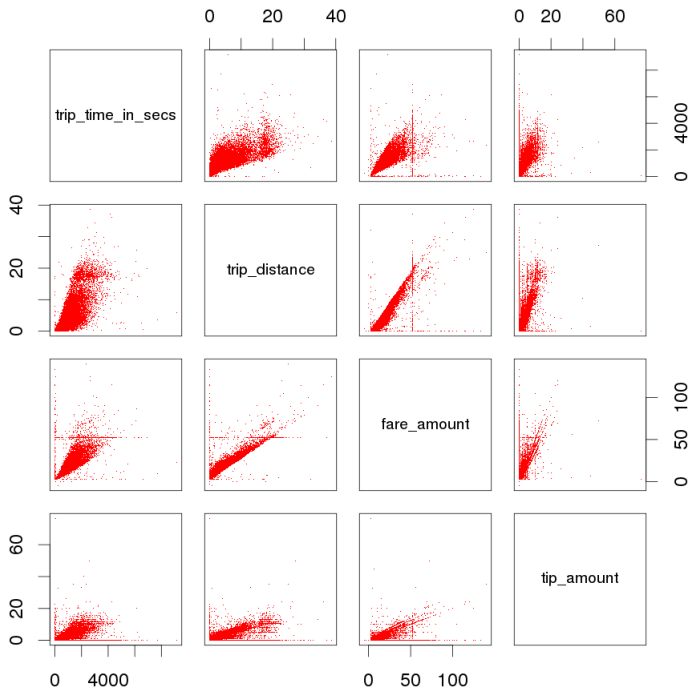
Table 1: Summary on fare and tip of 1% sample

fare amount		tip amount	
Min.	-5.00	Min.	0.000
1st Qu.	6.50	1st Qu.	0.000
Median	9.50	Median	1.000
Mean	12.53	Mean	1.366
3rd Qu.	14.50	3rd Qu.	2.000
Max.	229.00	Max.	76.500

The tabulated summary indicated that 1) the data quality is good in terms of the number of outliers, 2) the distribution of the variables in terms of inter-quartile ranges. In Section 2.4, we also summarized on the unsampled data set with Hadoop.

Visually, we used R's matrix plot to visualize relationships between variables (see Figure 1).

Figure 1: Matrix plot of trip duration, distance, fare and tip



The plot shows linear relationships between each pair of the four variables, trip duration, distance, fare and tip. Beyond the obvious linear relationships, we also discovered some strips in the tip versus fare plot. We suspected that these trips indicated tipping norms and we intended to verify it using the unsampled data set (see Section 2.6).

2.3 Data Filtering

Any "big data" data set has some erroneous records and the taxi data set was not an exception. Such errors might result from broken GPS devices, inaccurate driver-entered data (like the passenger count), and human errors in data compilation process. We identified such invalid records according to [7], [6], and [5]. For example, tips that exceeded 200 dollars were considered as invalid. For the geographical information like latitude and longitude, we restricted them inside longitude (-74.05, -73.70) and latitude

(40.60, 40.85). Note that they were not the actual city boundary, but they suffice for our analysis.

In our implementation, the filtering happened in two phases. The first filter was inside the record reader object, marking invalid fields and records as "NA" values. As it was in the record reader, the filter was mandatory. The second filter was implemented as a mapper. It removed any record with an "NA" field. This filter was used in all our analysis except in Section 2.4.

2.4 Assorted Statistical Summaries

Moving from sample data to the real data, we used Hadoop to facilitate processing speed. The first thing is to reproduce a similar summary report without sampling. We employed the Hadoop summarization pattern. Taking mean value calculation for example, in the mappers, we needed to keep track of per-mapper summation and record number, and in the singleton reducer, we combined the partial results and thus get the global mean value.

Unlike the summary report in R, we used standard deviation as a measure of distribution. It is commonly used to measure the spread of a variable. The definitive equation is in Equation 1.

$$var(X) = \sqrt{E(X - E(X))} \tag{1}$$

Using this equation, we need to separate the calculation into two MapReduce jobs, the first to calculate the mean, the second to calculate the standard deviation. As this calculation is not complicated and the process is obviously IO-bond, we intended to save the IO time by squeezing the calculation in one MapReduce job. We used the alternative standard deviation formula as shown in 2.

$$var(X) = \sqrt{E(X^2) - E(X)^2} \tag{2}$$

Adopting this formula, our program kept track of the summation, the summation of squared values, and the total number of records in each mapper. In this way the standard

deviation can be calculated in one pass. The result is shown in Table 2.

Table 2: Numerical variable summary

column	min	max	average	std dev	missing percent
trip distance	0.00	500.00	2.96	3.49	0.0000
trip time in secs	0.00	17820.00	767.98	575.94	0.0001
passenger count	0.00	8.00	1.70	1.37	0.0000
dropoff latitude	40.60	40.85	40.75	0.03	0.0201
dropoff longitude	-74.05	-73.70	-73.97	0.03	0.0168
mta tax	0.00	0.50	0.50	0.03	0.0001
fare amount	0.00	500.00	12.56	10.27	0.0001
tip amount	0.00	200.00	1.38	2.21	0.0000
surcharge	0.00	1.50	0.32	0.36	0.0000
pickup longitude	-74.05	-73.70	-73.98	0.04	0.0141
pickup latitude	40.60	40.85	40.75	0.02	0.0143
tolls amount	0.00	151.99	0.26	1.24	0.0000
total amount	0.00	699.96	15.02	12.32	0.0001

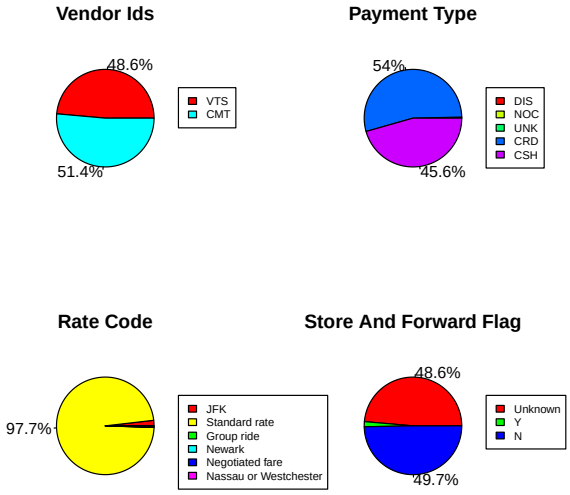
The result was consistent with our initial exploration with R. For most people, taxi was a short-distance transportation method. The outliers in each field does not exceed 2% of the records, indicating the reliability of the data. Note especially the total records was 54 million, but in this and some other calculations, we got 91 million. The reason is that we customized the input data format without paying attention to the unsplittable gzip format, resulting in 2 mappers reading some overlapped portions of the same file. As we write this article, the bug has been fixed, thanks to [10].

For categorical variables, we summarized the frequency of each level using Hadoop word count pattern. The result is shown in Figure 2. The figure indicated that 1) the two venders had similar market shares; 2) people preferred to pay by credit card; 3) most trips were in downtown, followed by trips from and to JKF airport.

2.5 Correlations and Dependency

Based on our initial exploration in Section 2.2, we wanted to confirm that some variables have linear correlations. For numerical variables, we used Pearson correlation coefficients. It is defined in Equation 3. Again, computation was less demanding than IO. So it was sensible to use an equivalent equation (as described in Equation 4) to fit the computation

Figure 2: Categorical variable summary



in one MapReduce job.

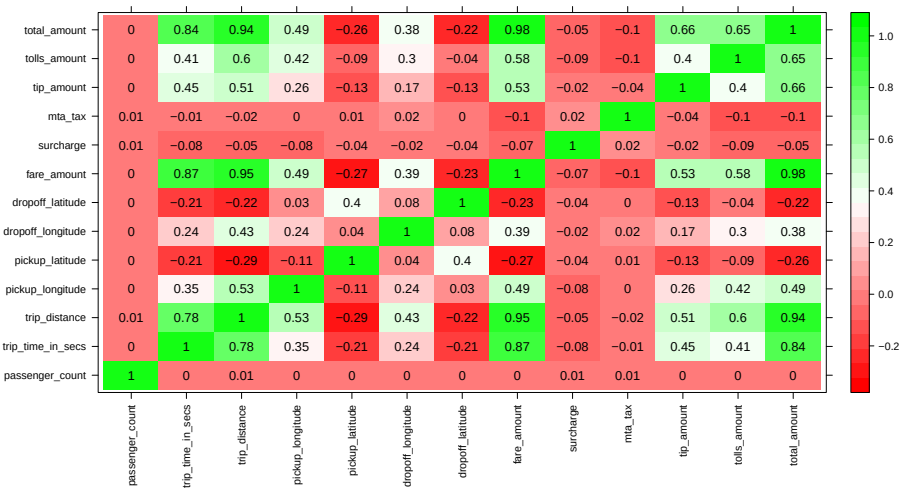
$$\rho = \frac{E(x - \bar{x})(y - \bar{y})}{\sigma_x \sigma_y} \tag{3}$$

$$\rho = \frac{E(x)E(y) - E(xy)}{\sqrt{E(x^2) - E(x)^2}\sqrt{E(y^2) - E(y)^2}} \tag{4}$$

The results shown in Figure 3 indicates that trip fares were a possible primary factor that affected how people pay tips. Surprisingly, the correlation coefficient of tip and fare is not close to 1. This might be caused by not filtering out transactions with cash as its payment type, where tips were not recorded and were always 0.

A typical statistical test of independence is called χ^2 test. The calculation includes building up a contingency table, calculating the expected values and then get the sum of squared error[11]. The resulting value is then compared with a table to determine dependence. In Hadoop, the challenge was to construct the contingency table. We used the structural pattern. The table was implemented as a data structure and each mapper used this structure as output. Then the reducer combined tables from mappers and

Figure 3: Correlation matrix



calculated the χ^2 value. The result is shown in Table 3. All these value is greater than the 5% confidence reference value.

Table 3: χ^2 test of independence

vender id & rate code	2.37E4
payment type & rate code	7.64E5
store fwd flag & rate code	3.45E4
payment type & store fwd flag	3.19E5
vender id & payment type	3.15E5
vender id & store fwd flag	8.9E7

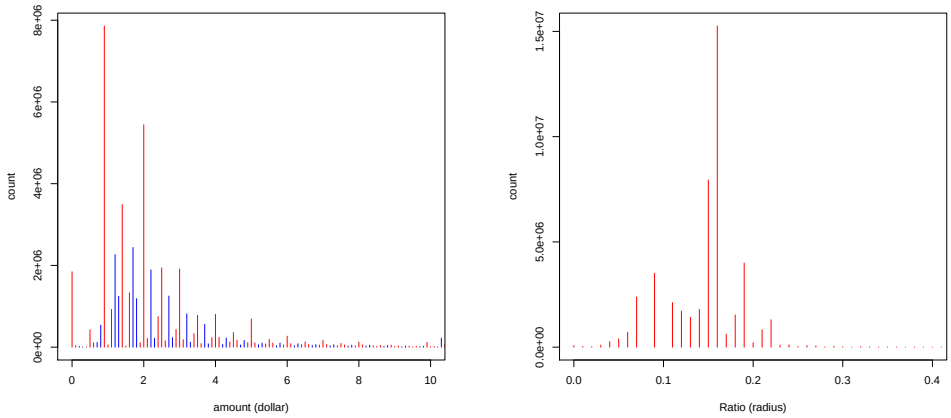
2.6 Interesting Tipping Behavior

In the following analysis, we filtered out all records with payment type being cash, because no tip was recorded. We built histograms of tip amount, tip ratios, locations, and time. After building a histogram of time, We also analyzed the temporal pattern using autocorrelation. The reason to use histogram was the continuous nature of the ratios, time and locations. Hadoop binning pattern was used.

The bin width for tip amount was 0.01 dollar. The bin width for ratio was 0.01 radius. The results are shown in Figure 5(a) and 5(b). Clearly, people tended to tip at integer amount, followed by amount that was multiple of 0.5 and 0.25 dollar, and at 20%, 10%, and 15%. Note that some gaps and peaks were caused by the inaccuracy of floating point

values. However, the overall distribution was not affected.

Figure 4: Interesting Tipping Behaviors

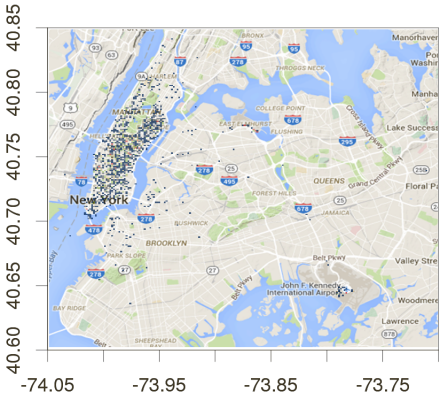


(a) Tip amount histogram

(b) Histogram of tip ratio

Average tips were binned according locations. The bin size was chosen so that the New York city was divided into 1000x1000 pixels. Hadoop Top-K pattern was used to select the top 100 locations with most average tips. The result is illustrated in Figure 5. Highest average tips were concentrated around the downtown area and the JFK airport.

Figure 5: Location with most tip



The trip data set contained pickup time. Thus we could use some common techniques in dealing with time series. A basic attribute is seasonality. To find seasonality, auto-correlation is commonly used. Its strict definition is described in Equation 5. We used a

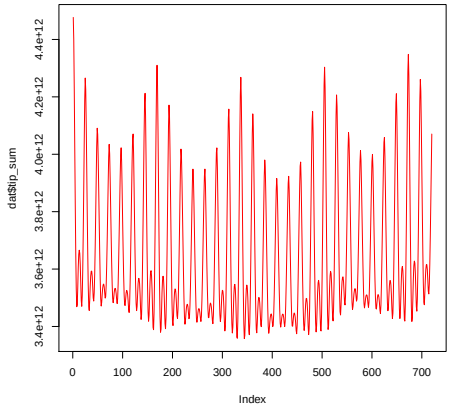
simpler definition (see Equation 6) that was commonly used in signal processing areas.

$$\rho_{xx}(\tau) = \frac{E(x_t - E(x))(x_{t+\tau} - E(x))}{\sigma_x^2} \tag{5}$$

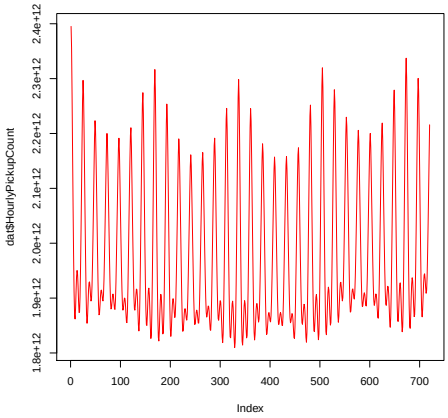
$$\rho_{xx}(\tau) = \sum x_t x_{t+\tau} \tag{6}$$

This task was separated into two MapReduce jobs. The first binned tip amounts into hours. The second did the calculation. Again, we used structural pattern so that a reducer could receive a variable and time delayed versions of itself. The results are shown in Figure 6.

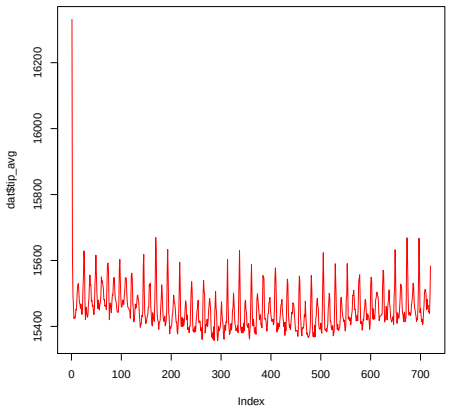
Figure 6: Seasonality of tipping behavior



(a) Autocorrelation of hourly tip sum



(b) Autocorrelation of hourly trip count



(c) Autocorrelation of hourly tip average

The plots indicate that evening to midnight was the busiest hour, especially in weekends. Drivers could receive more tip mostly because there were more business. The average tip amountm, however, did not show clear seasonality.

2.7 K-nearest-neighbor Tip Predictor

One of the simplest learning algorithm is call K-nearest-neighbor. The algorithm predicts unknown values by comparing the input with known instances. The algorithm requires large amount of data and computing power. However, it suits well on Hadoop. In our implementation of the algorithm, we first calculated the distance between the input and each record, then Top-K pattern was employed to efficiently choose the K nearest neighbors. To predict a numerical value, we needed to calculate the average of the K values. Because K was normally a small value, we used a python script to programmatically retrieve the top K values and calculate the average. The command-line application takes into account pickup, dropoff Lat-lon coordination, vender, day of week, hour of day, and rate code (type of trip).

3 Performance

Our execution environment was on a cluster of at least 36 nodes. All the nodes were virtual machines on a 2.4GHz 8-core Intel Xeon E5530. Each job spawned 36 mappers, one or up to 24 reducers, according to the computation need. The Hadoop block size was configured as 128MB. However, we used a unsplittable compression format (gzip). Thus each mapper read a whole file, or about 180MB. Because of a bug (described in Section 2.4), each input file was read nearly twice. The observed performance of is illustrated below (see Table 4).

4 Discussion

The results clearly indicates that tipping behaviors are highly influenced by temporal, spacial and psychological factors. However, the results are not comprehensive. As is true

Table 4: Performance

Job	execution time
Numerical Summary	1:12
Factor Summary	1:48
Correlation	1:40
χ^2	1:58
Autocorrelation	2:40
KNN	5:12

of any statistical investigation, the results are limited in terms of causality. For example, the valleys in hourly pick up count may be caused not by the demand for taxi services, but by congested traffic conditions.

Future analysis of the data set can split the data into more granular pieces and use ANOVA test to determine the dependence between categorical variables and numerical variables. To better understand tipping behavior, researches can focus on categorizing different tipping patterns based on regions, time, venders and etc.

The KNN predictor should only be used as a guidance. The model has not been validated thoroughly. Based on our manual validation, the predictor can sometimes give noisy results, because the learning data set has only gone through basic outlier removal only and KNN in particular is not resistive to noisy data. Besides, the KNN predictor performed badly in regions with sparse data.

Future implementation of the data set can first add weighting factors based on distances or regional norms. Further, a to achieve more understanding of tipping behaviors and more precise prediction, one can use the regression tree algorithm.

We hope that this article cast light on high performance statistical calculations with Hadoop.

References

[1] N. Taxi and L. Commission. (2015). Research and statistics, [Online]. Available: http://www.nyc.gov/html/tlc/html/about/statistics.shtml (visited on 10/2015).

[2] C. Whong. (2014). NYC taxis: A day in the life - a data visualization, [Online]. Available: http://www.nyc.gov/html/tlc/html/about/statistics.shtml (visited on 10/2015).

[3] Imagework. (). NYC taxi holiday visualization from JFK and LGA airports, [Online]. Available: http://taxi.imagework.com/ (visited on 10/2015).

[4] C. Ratti, M. Szell, E. Baczuk, B. Grob, J. Lee, P. Thebault, N. N. Lee, Y. J. Turgeman, A. Weib, and S. Landsbek. (). Exploring new york city taxi trails and sharing our way to a more sustainable urban future, [Online]. Available: http://hubcab.org/#11.00/40.7250/-73.8500 (visited on 10/2015).

[5] R. Hafen. (2015). Exploration of NYC taxi data, [Online]. Available: http://hafen.github.io/taxi/#initial-exploration (visited on 10/2015).

[6] A. James, "New York city taxicab transportation demand modeling for the analysis of ridesharing and autonomous taxi systems," 2015. [Online]. Available: http://orfe.princeton.edu/~alaink/Theses/SeniorTheses%2715/AJSwoboda--NYC_aTaxiRideSharingPotential2015.pdf (visited on 10/2015).

[7] NYC Taxi and Limousine Commission. (2015). Data dictionary - yellow taxi trip records, [Online]. Available: http://www.nyc.gov/html/tlc/downloads/pdf/data_dictionary_trip_records_yellow.pdf (visited on 10/2015).

[8] L. Zhou and J. Zhu. (2015). Csc582-1fa, [Online]. Available: https://github.com/lingyanzhou/CSC582-1FA (visited on 10/2015).

[9] acm. (2010). Apache hadoop: Best practices and anti-patterns, [Online]. Available: https://developer.yahoo.com/blogs/hadoop/apache-hadoop-best-practices-anti-patterns-465.html (visited on 10/2015).

[10] Apache Software Foundation. (). Hadoop core, [Online]. Available: http://grepcode.com/file/repository.cloudera.com/content/repositories/releases/com.cloudera.hadoop/hadoop-core/0.20.2-737/org/apache/hadoop/mapreduce/lib/input/TextInputFormat.java?av=f (visited on 10/2015).

[11] Wikipedia. (2015). Chi-squared test, [Online]. Available: https://en.wikipedia.org/wiki/Chi-squared_test (visited on 11/2015).