Redis & MongoDB Assignment

Team : Ling Oh Miao - f2821820 | Sarac Burcin - f2821825 | Schoch Piermattia (.. soon)

Scenario

We are a team of 3 analysts in a consulting firm and were tasked to perform an analysis on data related to classified ads from the used motorcycles market.

The aim of this report is to help our client to measure the effectiveness of the email solicitation method and also to understand the target market of classified advertisements from the used motorcycles market.

Our client has given us the access to two datasets:

- *Bikes_Datasets* includes details of items listed in the classified ads platform
- *Recorded_Actions* refers to seller's usage behaviour after they received a personalized e-mail to some of the sellers with a number of suggestions on how they could improve their listings.

Our team decided to separate the tasks into 2 parts and we would like to start analyzing the recorded actions using Redis and the details of the listings in MongoDB.

Task1: Recorded Actions

Approach

Due to academic nature of this project, Redis server was started without any explicit configuration file, so all the parameter will use the internal default. That means that server is configured to work on localhost at the port "6379".

Redis it is actually a data structures server, supporting different kinds of values. What this means is that, while in traditional key-value stores you associated string keys to string values, in Redis the value is not limited to a simple string, but can also hold more complex data structures. In this project Bit arrays (or simply bitmaps) seems to best choice to cope with this task. This data structure permits to handle String values like an array of bits.

One of the biggest advantages of bitmaps is that they often provide extreme space savings when storing information. For example in a system where different users are represented by incremental user IDs, it is possible to remember a single bit information (for example, knowing whether a user wants to receive a newsletter) of 4 billion of users using just 512 MB of memory (the maximum length of a string).

Starting from that, we replicate a real case scenario, using a dataset where we have datas about 19.999 users. Each question has been solved in the statistical software R, by "looping" over each row of the dataframes considering different combination of conditions. We then created the Bitmaps, through the command "SETBIT", setting the value of the "keys" = 1 if the conditions checked were satisfied. After that, in order to operate on group of bits we used two different commands:

1. "BITOP" performs bit-wise operations (AND, OR, XOR and NOT) between different strings.
2. "BITCOUNT" performs population counting, reporting the number of bits set to 1.

Connection to the local instance of Redis

```r
# Libraries
library(redux)

# Create a connection to the local instance of REDIS
r <- redux::hiredis(
            redux::redis_config(
             host = "127.0.0.1",
             port = "6379"))
```

1.1 How many users modified their listing on January?

9969

1.2a How many users did NOT modify their listing on January?

10031

1.2b Do these numbers match the total of your users?

We would like to highlight that the numbers does not adds up to the total of the users because for bit (and normal string) operations, Redis stores the length of the underlying allocated string at the byte level (1 byte = 8 bits). Redis has no way to store the exact bits length, so we get byte-sized return values. That means that if we have 19.999 users on the website and that each user has also a unique identifier, by setting the bits corresponding to the user ID that satisfy a particular key (in this case not modify their listing), Redis stores this info as 00010011………1, with a total lenght of 20000bits.

Since we calculate the total user that did not modify their listing through the bitwise operation "NOT" (which perform logical negation on each bit) using as a "searchkey" the one defined in question1, the last bit stored in that key from 0 becomes 1, incrementin by 1 the real number of user that did not modify the listing on January.

1.3 How many users received at least one e-mail per month (at least one e-mail in January and at least one e-mail in February and at least one e-mail in March)?

2668

4: How many users received an e-mail on January and March but NOT on February?

2417

5: How many users received an e-mail on January that they did not open but they updated their listing anyway?

2807

6: How many users received an e-mail on January that they did not open but they updated their listing anyway on January OR they received an e-mail on February that they did not open but they updated their listing anyway on February OR they received an e-mail on March that they did not open but they updated their listing anyway on March?

7221

7 Does it make any sense to keep sending e-mails with recommendations to sellers? Does this strategy really work? How would you describe this in terms a business person would understand?

In order to understad if the strategy is giving positive results, it is necessary to study the users' behaviour according to each action that they could do when they receive an email. More specifically we analyzed, with a monthly temporal window,4 different scenarios:

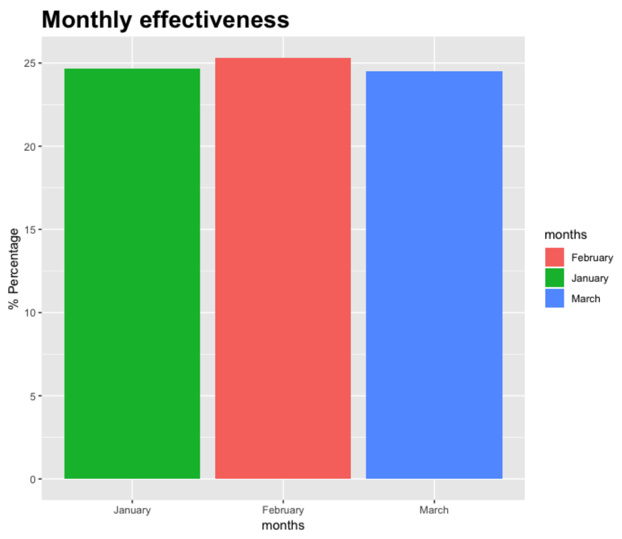
1. Green bar: Email Read , Modification
2. Red bar: Email Read , Non Modification
3. Salmon bar: Email non Read , Modification
4. Yellow bar: Email Non Read , Non Modification

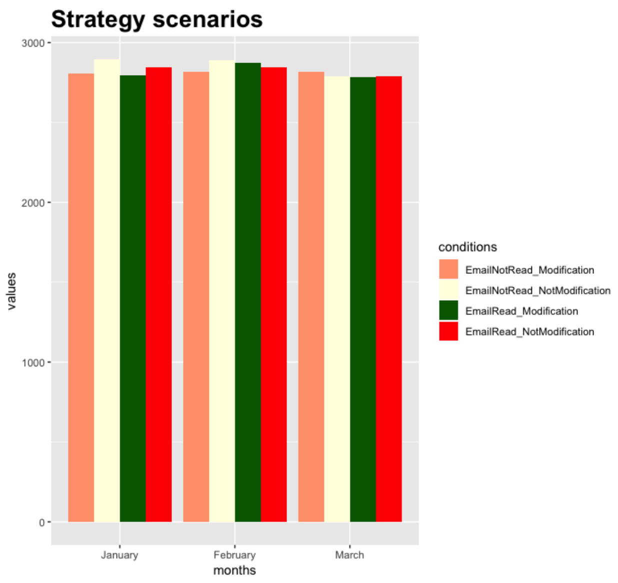
The result obtained, illustrated in the graphs below, doesn't show a positive response of the users from the email service.

The graph on the left illustrate the overall percentange result of the strategy in each month. Roughly 25% of the users responded positively to the personalized email campaign, in the sense that they performed a modification after reading the email. More specifically the percentange are 24.64, 25.32, 24.52 in each month.

The second graph on the right provide more details about users' behaviour. The green bar shows the frequencies of users that made a modification on their listing after the personalized suggestions. In order to consider the strategy implemented as a success, this bar should be considerably higher than the red one, which display the total amount of user that after reading the email, decided not to update their listing. The information depicted in the yellow and salmon bars strengthens the uncaring behaviour of the users towards this service provided.

To summarize, roughly 1/4 of the emails induced the users to modify their listing. That suggest that a change in the strategy should be taken in consideration, but it's necessary to have more details about the effort / cost of that operation.





Task2: Bikes

Approach

Bikes dataset consist on ~30K json files that describe classified ads from the used motorcycles market. To manage this amount of document we exploited MongoDB technology, which is a popular open-source document database for storing and manipulating very big JSON structures.

The first thing to do is to create a connection with the server: we run mongo shell without any command-line options to connect to a MongoDB instance running on our localhost with default port 27017. Rather than working on the shell, we decided to develop our code following two different approaches:

1. We magaged and queried our data directly in R, throught the use of two packages "jsonlite" and "mongolite". The former provides a powerful JSON parser and generator that has become one of standard methods for getting data in and out of R, the latter aims to provide a simple R client for MongoDB, simple meaning that you can insert and query data in R using data-frames with a single command.
2. We used RoboMongo, which is a visual tool helping you manage Database MongoDB (Open source project), that gave us a better interpretability on how to store, manipulate and query the documents.

2.1 Add your data to MongoDB.

To work with mongolite is necessary to operate with dataframes. The following steps aims to create and clean the dataset which will be later inserted in MongoDB. Once this step is completed it is possible to query the data.

Our solution consist in the following steps :

1. Create a list with all the paths of the files
2. Insert these files in MongoDb
3. Create a dataframe using mongolite command "findAll". This will crate a nested dataframe, since json documents are hierarchical.
4. Looking the structure of our json documents using SublimeText editor. Then create small dataframes (one for each key that defines a full records) and one that store the simple keys, then merge them all.
5. Cleaning operation with a small subset of the dataset origined in point5, then repeat this step with the original dataframe
6. Drop the old collection, and insert the new one in MongoDb.

In the next page we appended the code for the whole procedure according to the 6 points previuous mentioned.

```r
# ! Libraries !
library(jsonlite)
library(mongolite)
library(dplyr)

# -> 1. Reading Paths <-

# Import a vector containing all the paths
path = dir(path = "/Users/University/projects/BigDataSystem/BIKES", pattern = "\\.json$",
           full.names = TRUE, recursive = TRUE)

# Put as a list
pathlist = as.list(path)
#-------------------------
# -> 2-3. Insert in MongoDb and create a nested dataframe <-

# Open connection
m <- mongo(collection = "fulldf",  db = "Scrape_Bikes", url = "mongodb://localhost")

# Read each Json files and insert in the database
for(i in  1:length(pathlist)){
  m$insert(fromJSON(readLines(pathlist[[i]], warn = F)))
}

# Create a big dataframe (is composed by other dataframe (list in a list,
#                         nested dataframe))
newdf = m$find("{}")

# Look how nested is the structure
glimpse(newdf)

#----------------------------
# -> 4. Create small dataframes  <-

# Take dataframes
query_df = newdf[,"query"]
ad_data_df = newdf[,"ad_data"]
ad_seller_df = newdf[,"ad_seller"]
meta_data_df  = newdf[,"metadata"]
# Take Lists
list_df = newdf[,c("title","ad_id","extras","description")]

# Note there are two columns with the same name (type):
# MongoDB doesn't allow to have two keys with the same name (they are in query_df, meta_data_df)
colnames(meta_data_df) = c("type2","brand","model")

# Build the dataframe
trial = cbind(query_df, ad_data_df,ad_seller_df,meta_data_df,list_df)
```

```r
# -> 5. Cleaning  <-

# ! 1.PRICE !
# Remove € symbol
trial$Price <- gsub("€", "",trial$Price)
# You may need to escape the . which is a special character that means "any character"
trial$Price <- gsub("\\.", "",trial$Price)
# Transform in numeric
trial$Price = as.numeric(trial$Price)

# ! 2.MILEAGE !
# Remove Km
trial$Mileage <- gsub(" km", "",trial$Mileage)
# Remove commas
trial$Mileage <- gsub("," , "",trial$Mileage)
# Transform in numeric
trial$Mileage = as.numeric(trial$Mileage)

# ! 3.REGISTRATION ! (take care just on the year)
# Remove all before and up to ":":
trial$Registration = gsub(".*/","",trial$Registration)
# Transform in numeric
trial$Registration = as.numeric(trial$Registration)
#----------------------------
# -> 6. Insert cleaned dataset  <-

# Drop the previous collection
m$drop()
# Insert a new collection
t <- mongo(collection = "Cleaned",  db = "Scrape_Bikes", url = "mongodb://localhost")
t$insert(trial)
```

Once these step are done, we can inspect the structure of the dataframe obtained, by the function "glimpse" in the package "dplyr".

```r
# Look at the final dataframe
library(dplyr)
glimpse(trial)
```

This showed that we successfully extrapolate the information in each sub-dataframe and that we set properly a dataframe (each column is now a list) which can be work with "mongolite" package.

2.2 How many bikes are there for sale?

29701

2.3 What is the average price of a motorcycle (give a number)? What is the number of listings that were used in order to calculate this average (give a number as well)? Is the number of listings used the same as the answer in 1.2? Why?

The average price is 2962.701€. In order to calculate this quantity, 29145 records has been used. This is because there 556 records stored the price with a string "AskForprice".

It's important to notice that many listings have a symbolic price. Some sellers, in the second-hand market, prefer to choose this strategy simply because it stimulates curiosity, as it open a possible negotiation on the price. Considering the website scraped ("www.car.ge") and analyzing the market, it has been possible to set a threshold under which we have identified all the listing with a symbolic price. Even if it is arbitrary, we decided to set it to 70€: all the offers under this price are likely to include replacement parts or are symbolic. Generally from 70€ to 200€ all the listigs include crashed or very old and cheap motorbikes, so we decided to keep them all in the analysis.

Hence, excluding all this ads (1296), the average Price is 3030.31€.

2.4 What is the maximum and minimum price of a motorcycle currently available in the market?

The maximum price is 89.000€, while the minimum is 1€

2.5 How many listings have a price that is identified as negotiable?

There are 1348 listings that has been identified as "Negotiable"

2.6 For each Brand, what percentage of its listings is listed as negotiable?

Due to the long list of results, we appended this result in the appendix.

2.7 What is the motorcycle brand with the highest average price?

The motorcycle brand with the highest average price is "Semong" with an average price of 15600€.

2.8 What are the TOP 10 models with the highest average age? (Round age by one decimal number)

The top 10 models with the highest average age are illustrated in the table below. We found that there are 6 different names for the model “Bsa ’39”. Checking on internet and on the website scraped, we come up that they refers to the same model produce in the year 1939. In addition, since each model has been advertised only time (or in “Bsa” case 6 times with the same year), calculating the round average value is useless.

	Models	AverageAge
1	Allo henderson indian replica ’31	88.00
2	R 12	85.00
3	Norton ’35	84.00
4	Allo MATCHLESS G3 350 ’35	84.00
5	Norton H16 ’36	83.00
6	Allo Matsoules ’38	81.00
7	Bsa 20 ’39	80.00
8	Allo Matchless G3L	80.00
9	Allo NEW HUDSON ’39	80.00
10	Allo Matsoules1 ’40	79.00

2.9 How many bikes have “ABS” as an extra?

There are 4025 bikes that have “ABS” as an extra

2.10 What is the average Mileage of bikes that have “ABS” AND “Led lights” as an extra?

1135km.

2.11 What are the TOP 3 colors per bike category?

We obtained, for categories that are not listed frequently, an equal value for the last most frequent color. The full results with the count for each color are in the R script.

	Category	Top3 Colors
1	Bike - Trial	Red, White, Black (Metallic)
2	Bike - Mobility scooter	Red, Grey, Black (Metallic)
3	Bike - UTV Side by Side	Black, Blue, White
4	Bike - Buggy	Black , Red , Blue
5	Bike - Cafe Racer	Black, Black (Metallic), Red,
6	Bike - Moped	Red, Black, Blue,
7	Bike - Four Wheel-ATV	Red, Black, White
8	Bike - Custom	Black, Black (Metallic), ”Red
9	Bike - Mini..Moto	Red, Black, ”White)
10	Bike - Naked	Black, Black (Metallic), White (Metallic)
11	Bike - Motocross	Red, Orange, Green
12	Bike - On/Off	Black, Black (Metallic), White
13	Bike - Super Sport	Black, Black (Metallic), Red
14	Bike - Chopper	Black, Black (Metallic), Bordeaux (Metallic)
15	Bike - Enduro	White, Orange, Red,
16	Bike - Super Motard	Black, Orange, Black (Metallic),
17	Bike - Three Wheel	Black, White, Red)
18	Bike - Sport Touring	Black (Metallic), Black, Silver (Metallic)
19	Bike - Underbone	Black, Blue, Black (Metallic)
20	Bike - Street Bike	Black, Black (Metallic), Red
21	Bike - Other	Black, Black (Metallic), Red
22	Bike - Roller/Scooter	Black, Black (Metallic), White

2.12 Identify a sets of ads that you consider "Best deals"

In order to find the best deals, different arbitrary criteria could be used. Considering the assigned task, we filtered the dataset according to general rules that a professional company operating in this sector should be take in consideration. Our final solution consist in 411 ads, which are all good deal.

In the list we show the criteria used to find the good deals.

1. *Profitable Ads* : Lower price than the average price of that category of bikes:
2. *Good Engine*: Lower mileage than the average mileage of that brand (we grouped per brand because it's implicit that some brand have a more powerful engine, that permits an higher engine life)
3. *Relatively New* : Not more old than 2014.
4. *Official Dealer*: We prefer to suggest offers by professional dealer, rather than private seller, since they offer higher warranty protection
5. *Real Price* : We deleted from our dataset all the symbolic prices, even if it is higly likely that the ads from official dealer should not contained them.

The query below , can be easily modified to meet different specific needs.

```
#---------------------- > FILTERING BEST DEALS < ---------------
# Query
pipe = '[
    {"$project": { "Classified number" : 1,
                   "url" : 1 ,
                   "age" : 1,
                   "Price": 1,
                   "Previous owners" : 1,
                   "Dealer" : 1,
                   "Category" : 1,
                   "low_price_brand":{"$cmp": ["$Price","$avgPricebyBrand"]} ,
                   "low_mileage": {"$cmp": ["$Mileage", "$avgMileagebyBrand"]},
                   "low_price_cat" : {"$cmp": ["$Price", "$avgPricebyCategory"]}}},
    {"$match": {
        "$and":[
                {"low_mileage":{"$lt":0}},                   # Lower mileage
                {"low_price_cat":{"$lt": 0}},                # Lower price category
                {"age" :{"$lt" :5}},                         # 2014-2019
                {"Price" : {"$gte" : 250}},                  # Not symbolice
                {"Dealer": {"$exists":true, "$ne" : "NA"}}   # Official dealer
                ]
     }
    }
  ]'

# Dataframe containing url of best deals
bestdeal = t$aggregate(pipeline = pipe)
```

Appendix

2.6 Results

In the next pages it's possible to see the percentange of listing listed as negotiable, per brand.

brand	Negotiable Listing	Total Listing	Percentage
Joyner	2	2	100 %
Garelli	6	48	12.5 %
Super Moto	2	4	50 %
E-Ton	2	4	50 %
Kuberg	2	2	100 %
Bajaj	4	12	33.3 %
Jinlun	2	2	100 %
Zuendapp	6	14	42.9 %
Tgb	2	14	14.3 %
Regal-Raptor	4	4	100 %
Adiva	8	10	80 %
Vmoto	2	12	16.7 %
Gemini	8	12	66.7 %
Simson	6	22	27.3 %
Lem	6	24	25 %
AGM motors	2	10	20 %
CPI	2	20	10 %
JetMoto	4	8	50 %
Kxd	2	14	14.3 %
Nipponia	2	18	11.1 %
Skyteam	2	88	2.3 %
Mtg	8	8	100 %
Kinroad	2	8	25 %
Baotian	16	28	57.1 %
Motobi	2	2	100 %
Swm	4	12	33.3 %
Maico	2	8	25 %
Dias	6	6	100 %
Lintex	2	6	33.3 %
Modenas	20	522	3.8 %
Nomik	2	4	50 %
Access Motor	4	14	28.6 %
Gilera	36	1180	3.1 %
SMC	6	18	33.3 %
Dayang	10	60	16.7 %
Yamaha	312	11058	2.8 %
HighPer	4	4	100 %
Cheetah	12	34	35.3 %
Qingqi	4	4	100 %
Honda	494	12380	4 %
Peugeot	12	612	2 %

ZhongYu	2	2	100 %
Dirt Motos	12	30	40 %
Hercules	2	6	33.3 %
Xgjao	14	16	87.5 %
Jincheng	4	14	28.6 %
MBK	4	10	40 %
Haojin	14	20	70 %
Generic	2	12	16.7 %
MZ	6	22	27.3 %
Triumph	12	670	1.8 %
Niu	2	2	100 %
Benelli	6	82	7.3 %
Cagiva	6	64	9.4 %
Jialing	2	40	5 %
Ymc	16	72	22.2 %
Linhai	4	96	4.2 %
Keeway	18	206	8.7 %
Derbi	10	172	5.8 %
Lifan	50	208	24 %
Amstrong	6	6	100 %
Nitro Motors	2	2	100 %
Kymco	42	2296	1.8 %
Rewaco	2	6	33.3 %
FB Mondial	4	8	50 %
Indian	4	6	66.7 %
E-ATV	4	4	100 %
LML	12	54	22.2 %
Jonway	6	10	60 %
Bombardier	2	2	100 %
Harley Davidson	24	618	3.9 %
CFmoto	6	42	14.3 %
Malaguti	4	110	3.6 %
Hyosung	10	86	11.6 %
Norton	2	10	20 %
Montesa	4	12	33.3 %
Beta	22	80	27.5 %
Horex	2	12	16.7 %
Sherco	2	2	100 %
Mv Agusta	4	64	6.2 %
Boom-Trikes	2	2	100 %
Kaisar	6	8	75 %
Quadro	4	10	40 %

Victory	2	2	100 %
Shineray	6	56	10.7 %
Ural	2	10	20 %
Odess	4	4	100 %
Puch	4	18	22.2 %
CAN-AM	4	44	9.1 %
Imr	8	20	40 %
Fuxin	2	8	25 %
Motivas	2	6	33.3 %
KTM	42	1932	2.2 %
Emw	2	4	50 %
Buggy Motors	6	6	100 %
Polaris	12	70	17.1 %
Ducati	12	746	1.6 %
Aeon	2	66	3 %
Fever	2	2	100 %
Solex	8	30	26.7 %
Bultaco	2	4	50 %
Asus	2	10	20 %
Husqvarna	12	290	4.1 %
Znen	8	12	66.7 %
Vespa	28	548	5.1 %
TCB	8	88	9.1 %
TM	12	66	18.2 %
Jmstar	2	2	100 %
Boatian	4	12	33.3 %
AB	16	26	61.5 %
Italjet	2	10	20 %
Skyjet	10	20	50 %
Moto Morini	2	8	25 %
Bsa	10	30	33.3 %
Dkw	18	32	56.2 %
Heinkel	4	10	40 %
Wsk	2	2	100 %
Daelim	2	138	1.4 %
Hsun	4	14	28.6 %
Moto Guzzi	2	100	2 %
Bmw	56	2788	2 %
Aprilia	32	1784	1.8 %
Kawasaki	130	3906	3.3 %
Vee Road	2	2	100 %
Zundapp	40	124	32.3 %

Siamoto	2	10	20 %
Kreidler	34	166	20.5 %
Daytona	104	786	13.2 %
Piaggio	90	5370	1.7 %
Sachs	38	124	30.6 %
Loncin	6	50	12 %
Royal Enfield	10	26	38.5 %
”ll”	242	678	35.7 %
Gas-Gas	6	70	8.6 %
Zongshen	10	52	19.2 %
Bashan	22	42	52.4 %
Arctic Cat	4	16	25 %
Apokotos	4	4	100 %
Husaberg	2	60	3.3 %
Lambretta	20	28	71.4 %
Pgo	4	40	10 %
Sym	40	2180	1.8 %
Jawa	14	32	43.8 %
Access	6	18	33.3 %
Suzuki	104	4730	2.2 %
Vor	2	12	16.7 %
Xingyue	6	16	37.5 %