Report on MediaTech Internship at NBCUniversal

Emi Harry

12/18/2017

Executive Summary

According to a publication by Accenture in 2015, 89% of the users they surveyed watched some sort of long video on their mobile devices.

AT&T claims to have had a 75% increase in data usage over the past year.

Users are more glued to their mobile devices than ever and its only set to increase.

This internship focused on the first exploration of the NBCUniversal WatchTower data, with the hope of gaining meaningful insights on the behavior of consumers.

Due to the sensitive nature of the consumer data, exact results will not be disclosed in this report.

Introduction

This MediaTech fall internship commenced in September 2017, ending in December 2017. The research work was carried out under the direct supervision of Hristina Galabova, Senior Director, Solutions Delivery, in the Digital Products and Interactive Media Department at NBCUniversal, 5 Times Square, New York.

Job Description

At the point of recruitment, my job description was listed as follows:

- Conduct analytics on existing consumer behavioral data.
- Processing of real consumer data and ability to apply tech analysis to driving product development.
- Produce meaningful results from a large set of data.
- Gathering insights into this yet untapped set of consumer data would help inform new product features to further drive consumer engagement.

Specific Project Description

Overview

The watchTower data is stored for the sole purpose of ensuring that the mobile app users are able to continue watching a specific video from where they last stopped.

Task

Since the data had never be analysed, I was tasked with carrying out analysis on it to see if there were useful insights within the data. Being given free rein on the technologies to use, as such, the programming language being used is R, while Tableau will be used for all visualizations.

Research Goals

- Understand the habits of users per brand.
- Proffer ways of increasing the number of digital users across brands.

Implementation

Data Structure

The data from watchTower was extracted into 3 different csv files, each being approximately 7GB in size. Having 8 features, with over 20 million rows each.

Stage 1: Data Cleaning and Pre-processing in R

The first step in the data cleaning process was to view the data in a simple text editor. This enabled me to understand the format of the data identify the following:

- Relevant data to be read into R
- characters like '{},“”,:' that needed to be removed from the data frame.

The sample of the raw data is as shown below:

id,flags,expiration,cas,value,rev,vbid,dtype

072b53d0-0cd5-46d9-abf3-d1c04b73f68c,33554432,0,1482372320294338560,

“{”“showId”“:”“7162758c-7540-4ef9-ada5-30215c8ba039”“,”“deleted”“:false,

“”favoriteType“”:“”shows“”,“”customFields“”:{“”device“”:“”IOS“”},

“”brandId“”:“”bravo“”,“”createdTimestamp“”:1482372320289,“”_class“”:

“”com.nbcuni.watchtower.api.model.Favorite“”,“”type“”:“”FAVORITES“”,

“”userId“”:“”b708e56537fa18aefb3c6180223a437138534f107e965c56b2e667562fb1dd79“”,

“”updatedTimestamp“”:1482372320289}“,3,818,1

Based on the sample above, I worte a function called “cleanWatchTower”, which takes in the file name and retruns a list consisting of two separate data frames. One for the favorites data, and the other for the watches data.

```r
# define the function to take in x which should be the file name if the file
# is in the current working directory and full directory link if it isn't.
cleanWatchTower <- function(x){
  library(data.table)
  library(plyr)
  library(dplyr)
  library(tidyr)
  library(stringr)
  library(sqldf)

  # read in the first data set
  data <- fread(x, select = c("value"))

  # separate watching data from favourites
```

```r
watching <- sqldf("select * from data where value LIKE '%percentViewed%'")
favourites <- sqldf("select * from data where value LIKE '%favoriteType%'")

# remove the original data
rm(data)

# remove all special characters from the favourites data

favourites[] <- lapply(favourites, gsub, pattern= 'customFields"":', replacement='')
favourites[] <- lapply(favourites, gsub, pattern= '"', replacement='')
favourites[] <- lapply(favourites, gsub, pattern= '[{}]', replacement='')
favourites[] <- lapply(favourites, gsub, pattern= ',', replacement=', ')
favourites[] <- lapply(favourites, gsub, pattern= ':', replacement=': ')

# separate the column in to multiple columns of different fields

newfav <- favourites %>%
  separate(value, c("showId", "deleted", "favoriteType", "device",
                    "brandId", "createdTimestamp", "class", "type", "userId",
                    "updatedTimestamp"), ",") %>%
  select(showId, deleted, brandId, createdTimestamp, type, userId,
         updatedTimestamp)

newfav[] <- lapply(newfav, gsub, pattern= ".*:", replacement='')


# remove all special characters from the watching data

watching[] <- lapply(watching, gsub, pattern= 'customFields"":', replacement='')
watching[] <- lapply(watching, gsub, pattern= '"', replacement='')
watching[] <- lapply(watching, gsub, pattern= '[{}]', replacement='')
watching[] <- lapply(watching, gsub, pattern= ',', replacement=', ')
watching[] <- lapply(watching, gsub, pattern= ':', replacement=': ')

# separate the column in to multiple columns of different fields

newwatch <- watching %>%
  separate(value, c("continueWatchingList", "watchDuration", "showName",
                    "seasonNumber", "device", "platform", "createdTimestamp",
                    "videoId", "type", "userId", "updatedTimestamp", "showId",
                    "deleted", "brandId", "percentViewed", "class", "mpxGUID",
                    "dateTimeWatched"), ",") %>%
  select(userId, showId, showName, brandId, mpxGUID, percentViewed, seasonNumber,
         watchDuration, deleted, createdTimestamp, updatedTimestamp, dateTimeWatched,
         type, continueWatchingList)

newwatch[] <- lapply(newwatch, gsub, pattern= ".*:", replacement='')

# remove all na

newwatch <- newwatch[complete.cases(newwatch), ]
newfav <- newfav[complete.cases(newfav), ]
```

```
  # convert the time and date from UNIX millisecond format to real time

  newfav$createdTimestamp <- as.POSIXct(as.numeric(as.character(newfav$createdTimestamp))
                                        /1000,origin="1970-01-01", tz = "UTC")
  newfav$updatedTimestamp <- as.POSIXct(as.numeric(as.character(newfav$updatedTimestamp))
                                        /1000,origin="1970-01-01", tz = "UTC")


  newwatch$createdTimestamp <- as.POSIXct(as.numeric
                                          (as.character(newwatch$createdTimestamp))/1000,
                                          origin="1970-01-01", tz = "UTC")
  newwatch$updatedTimestamp <- as.POSIXct(as.numeric
                                          (as.character(newwatch$updatedTimestamp))/1000,
                                          origin="1970-01-01", tz = "UTC")
  newwatch$dateTimeWatched <- as.POSIXct(as.numeric
                                         (as.character(newwatch$dateTimeWatched))/1000,
                                         origin="1970-01-01", tz = "UTC")

  # separate the date and time into individual columns

  newwatch <- newwatch %>%
    separate(createdTimestamp, c("dateCreated", "timeCreated"), " ") %>%
    separate(updatedTimestamp, c("dateUpdated", "timeUpdated"), " ") %>%
    separate(dateTimeWatched, c("dateWatched", "timeWatched"), " ")

  newfav <- newfav %>%
    separate(createdTimestamp, c("dateCreated", "timeCreated"), " ") %>%
    separate(updatedTimestamp, c("dateUpdated", "timeUpdated"), " ")

  # remove all na again

  newwatch <- newwatch[complete.cases(newwatch), ]
  newfav <- newfav[complete.cases(newfav), ]
  output <- list()
  output$newfav <- newfav
  output$newwatch <- newwatch

  return(output)

}
```

Next the function is used to call and clean the files, as shown below. Grouping them into favorites (the record of shows that have been marked as favorites by users), and watches (the record of show that a user was watching and its progress).

```
# apply the customized parsing function to each data set.
data1 <- cleanWatchTower("watchtower_NBCPCONRAPIWT_10.17.103.61%253A8091.csv")
newfav1 <- data1$newfav
newwatch1 <- data1$newwatch

data2 <- cleanWatchTower("watchtower_NBCPCONRAPIWT_10.17.103.125%253A8091.csv")
newfav2 <- data2$newfav
newwatch2 <- data2$newwatch
```

```
data3 <- cleanWatchTower("watchtower_NBCPCONRAPIWT_10.17.103.126%253A8091.csv")
newfav3 <- data3$newfav
newwatch3 <- data3$newwatch
```

Running on the assumption that if a show has been marked by a user as a favorite, then it implies that the user has definitely watched the show. Conversely, that a user watches a show does not necessarily mean they will mark it as a favorite.

As such, I wrote another function which utilizes an SQL query to join the watch data frame to the favorite data frame by their userId. Ensuring that only users found in the favorite are taken from the watch. A new column called 'favorites' is created and all users that have marked shows as favorites are depicted by 1 in that column. Next, the users frow the watch data frame that were omitted from the initial SQL join are extracted from the original watch data frame and are also given a new column 'favorites' with 0 assigned to all fields, depicting "not marked as favorite".

After this the function merges the '0' and '1' data frames, churning out the final result. The function implementation can be see below.

```
# define the function to take in the 2 data frames
compare_mutate <- function(x,y){
  library(sqldf)
  library(dplyr)
  x <- x[, -c(10,19,20, 21)]
  y <- y[, -c(3,8)]
  setnames(y, c("dateCreated", "time.Created", "millisecsCreated",
              "dateUpdated", "time.Updated", "millisecsUpdated",
              "unixtimeCreated","unixtimeUpdated"),
           c("F.dateCreated", "F.time.Created", "F.millisecsCreated",
             "F.dateUpdated", "F.time.Updated", "F.millisecsUpdated",
             "F.unixtimeCreated","F.unixtimeUpdated"))
  nudf <- sqldf("select * from x
              join y on x.showId == y.showId
              and x.userId == y.userId")
  nudf <- nudf[, -c(18,19,20,24)]
  nedf <- anti_join(x, nudf)
  nudf <- nudf %>%
    mutate(favorites = 1)
  nedf <- nedf %>%
    mutate(F.dateCreated = 0, F.time.Created = 0, F.millisecsCreated = 0,
           F.dateUpdated = 0, F.time.Updated = 0, F.millisecsUpdated = 0,
           F.unixtimeCreated = 0 ,F.unixtimeUpdated = 0, favorites = 0)
  df <- rbind(nudf, nedf)
  df <- df %>%
    arrange(userId)
  return(df)
}
```

Next the function is applied to the previously cleaned data, after which they are all combined into one data frame, as shown below.

```
# Combine the watch transcations and favorite transactions for the 3 different data sets
result1 <- compare_mutate(newwatch1, newfav1)
result2 <- compare_mutate(newwatch2, newfav2)
result3 <- compare_mutate(newwatch, newfav)

# Combine all 3 data sets and ensure there are no duplicates
```

```r
cl <- rbind(result1, result2)
cl <- rbind(cl, result3)
cl <- unique(cl)
```

Stage 2: Preliminary Analysis in R

During the first assessment of the data set, there were 2 callenges faced.

1. The files were too large: This was handled by the fread function from the dplyr package, which enabled me to read only the necessary data into R.
2. There were transactions in the data with duplicate values.Since the transaction Ids were not important in this research, that variable was ignored, allowing for the removal of all duplicates.

Stage 3: What Questions are We Trying to Answer?

After much brainstroming, the following questions came out on top.

- How many users are registered on more than one brand?
- How many usere watch more than one television show?
- At what time do users typically watch shows?
- How many users watch only the show in their favorites?

The following code below was used it carrying out the necessary computations to answer these questions.

```r
# count of brands per user
bT_ormore <- cl %>%
  select(userId, brandId, dateWatched, time.Watched, percentViewed) %>%
  group_by(userId) %>%
  distinct() %>%
  mutate(brand_count = n_distinct(brandId)) %>%
  arrange(userId)

# remove years that don't correspond with 2017 and 2016
not2017 <- bT_ormore %>%
  mutate(years = year(as.Date(dateWatched))) %>%
  filter(!(years <= 2017 & years >= 2016))
bT_ormore <-
ts <- not2017[,-7]
is2017 <- anti_join(bT_ormore, ts)
oneBrand <- Tu_ormore %>%
  filter(brand_count==1)
twoBrands <- Tu_ormore %>%
  filter(brand_count==2)
threeBrands <- Tu_ormore %>%
  filter(brand_count==3)

# count the number of shows per user
sT_ormore <- cl %>%
  select(userId, showName, dateWatched, time.Watched, percentViewed) %>%
  group_by(userId) %>%
  distinct() %>%
  mutate(show_count = n_distinct(showName)) %>%
  arrange(userId)

# remove years that don't correspond with 2017 and 2016
not2017 <- sT_ormore %>%
  mutate(years = year(as.Date(dateWatched))) %>%
  filter(!(years <= 2017 & years >= 2016))

ts <- not2017[,-7]
is2017 <- anti_join(sT_ormore, ts)
oneShow <- S_ormore %>%
  filter(show_count==1)
twoShows <- S_ormore %>%
  filter(show_count==2)
threeShows <- S_ormore %>%
  filter(show_count==3)

# create structured data frame of users and their shows
twoShows <- twoShows %>%
  group_by(userId) %>%
  mutate(what = c("show1", "show2"))
threeShows <- threeShows %>%
  group_by(userId) %>%
  mutate(what = c("show1", "show2", "show3"))
expeu <- twoShows %>%
  spread(what, showName)
tu_tally <- expeu %>%
  group_by(show1, show2) %>%
  tally()
expeu2 <- threeShows %>%
  spread(what, showName)
tre_tally <- expeu2 %>%
  group_by(show1, show2, show3) %>%
  tally() %>%
  arrange(n)

# create structured data frame of users and their brands
tworand <- twoBrands %>%
  group_by(userId) %>%
  mutate(what = c("brand1", "brand2"))
threerand <- threeBrands %>%
  group_by(userId) %>%
  mutate(what = c("brand1", "brand2", "brand3"))
expeu <- tworand %>%
  spread(what, brandId)
tu_tally <- expeu %>%
  group_by(brand1, brand2) %>%
  tally()
tu_abv20 <- tu_tally %>%
  filter(n>=20)
expeu2 <- threerand %>%
  spread(what, brandId)
tre_tally <- expeu2 %>%
  group_by(brand1, brand2, brand3) %>%
  tally() %>%
  arrange(n)
tre_abv20 <- tre_tally %>%
  filter(n>=20)

# manipulating the data to get it in the deaired structure
suw <- data.frame(t(apply(tre_tally, 1, sort)))
weee <- data.frame(matrix(0, nrow = 277, ncol = 4))
weee$X1 <- as.character(suw$X2)
weee$X2 <- as.character(suw$X3)
weee$X3 <- as.character(suw$X4)
weee$X4 <- as.character(suw$X1)

we2 <- data.frame(matrix(0, nrow = 4, ncol = 4))
we2$X1 <- as.character((weee[274:277, 4]))
we2$X2 <- as.character((weee[274:277, 1]))
we2$X3 <- as.character((weee[274:277, 2]))
we2$X4 <- as.character((weee[274:277, 3]))

weee[274:277, 1] <- we2[1:4, 1]
weee[274:277, 2] <- we2[1:4, 2]
weee[274:277, 3] <- we2[1:4, 3]
weee[274:277, 4] <- we2[1:4, 4]

suw <- weee %>%
  group_by(X1, X2, X3) %>%
  summarise(No_Users = sum(as.numeric(X4))) %>%
  unite(brands, X1, X2, X3, sep = ",") %>%
  filter(No_Users > 100)
write.csv(suw, "tri_brand.csv")
```

After running the preliminary analysis code above, a clear path was realized, having gained sufficient answers to the questions above, a final computiation was run using the code below. This enabled me to have the data in a consolidated format, allowing for easy visualization.

```
# filter out the years that dont match 2016 and 2017
clWT <- cl %>%
  mutate(years = year(as.Date(dateWatched))) %>%
  filter(years >= 2016  & years <= 2017) %>%
  select(-years) %>%
  distinct()

# merge the data with counts of shows and brands per user
sb <- merge(bT_ormore, sT_ormore, by = c("userId", "dateWatched", "time.Watched"))
sb <- unique(sb[, -c(2,3,5,8)])

# function to check the most frequent entry in a given field
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

# duplicate records with different transaction id
dup <- clWT[,-1]
```

```r
duplicates <- clWT[duplicated(dup),]

# count the number of entries per user
topusers <- clWT %>%
  select(userId, brandId, showName, mpxGUID, dateWatched, time.Watched, percentViewed) %>%
  group_by(userId) %>%
  distinct() %>%
  ungroup(userId) %>%
  add_count(userId)

# since add_count gives the new column the name "n", change the name
setnames(topusers, c("n"), c("record_count"))

# count the number of videos viewed per user and chang the column name as well
user_rank <- topusers %>%
  select(userId, brandId, showName, dateWatched, time.Watched, mpxGUID,
         percentViewed, record_count) %>%
  group_by(userId, showName) %>%
  mutate(videos_per_show = n_distinct(mpxGUID))
setnames(user_rank, c("n"), c("videos_per_show"))

# chang the datewatched to specific weekdays and the time to specific hour in
user_rank$dateWatched <- weekdays(as.Date(user_rank$dateWatched))
user_rank$time.Watched <- hour(as.POSIXct(user_rank$time.Watched, format = "%T",
                                          origin="1970-01-01"))

'
find the most frequent day, time, average videos per show  and average percentviewed,
by each user for each show they have watched.
'
user_analysis <- user_rank %>%
  select(-mpxGUID) %>%
  mutate(freq_day = Mode(dateWatched), freq_hour = Mode(time.Watched)) %>%
  select(-dateWatched, -time.Watched) %>%
  mutate(avg_viewcount = mean(videos_per_show),
         avg_percentviewed = mean(percentViewed)) %>%
  select(-percentViewed)
user_analysis <- unique(user_analysis)

userstats <- merge(user_analysis, sb, by= c("userId", "brandId", "showName"))
userstats <- unique(userstats)

# final data
newdata <- clWT[, -c(1,3,7:15,18:25)]
newdata <- unique(newdata)
data <- merge(newdata, userstats, by= c("userId", "brandId", "showName"))
data <- unique(data)

# remove esquire from the data set.
esquire <- data %>%
  filter(str_detect(brandId, "esquire"))
final <- anti_join(wt, esquire)
```

```
# filter out any duplicates
wt <-unique(final)

# export the final data as a csv for visualization in Tableau
write.csv(wt,"WT.csv")
```

Stage 4: Visualizations in Tableau

To better interpret the findings, the extreme user persona was created.

We have two types of extreme users, defined as follows:

Extreme User 1

- Users loyal to one brand.
- They watch multiple shows consistently.
- They watch dedicatedly at specific time frames.
- They always complete viewing the videos they start.

Extreme User 2

- Users with two or more brands.
- They watch multiple shows consistently.
- They watch dedicatedly at specific time frames.
- They always complete viewing the videos they start.

The link to the visuals is below

```
[click here] (https://us-east-1.online.tableau.com/t/visualizationsbyemiharry/views/part3
 /Story1?:embed=y&:showAppBanner=false&:showShareOptions=true&:display_count=no&:showVizHome=no)
```

Stage 4: Insights

- App Usage: There are brands with excessive content in comparison to the number of users.
- Favorites: The favorites function does not add to the user experience.
- Outliers: There are users who do not fit in to the standard pattern. These are the extreme users.

Recommendations and Conclusion

Recommendations

- Carry out user surveys, grouping them based on their usage patterns.
- Identify the motivation behind the usage pattern of extreme users
- Suggest shows to non-extreme users across brands based on the viewing patterns of users similar to them.
- Remove the favorites feature.

Conclusion

The field of big data analytics or data science is definitely industry agnostic. No matter have trival the data is, there always lies useful information. Maybe already discovered, maybe not. But from this reasearch project I have learned that it is always worth a shot to analyse the data you have as it may have vital information the business needs to succeed.