# Exploratory Data Analysis of 'AOL Logs'

*RAJ KUMAR*

*April 14, 2017*

### Data Challenge : AOL Search Logs

Clear Environment

```
rm(list=ls())
```

Loading packages needed

```
library(data.table)
library(stringi)
library(sqldf)
library(tm)
library(SnowballC)
library(wordcloud)
library(RColorBrewer)
library(qdap)
library(tm)
library(knitr)
library(rmarkdown)
```

Reading Data into R where fields are tab separated

```
df<-read.csv("/home/raj/Downloads/C1X/DC/user-ct-test-collection-02.txt",
             header = TRUE,sep="\t")
```

Converting to data table

```
df<-as.data.table(df)
```

Data Cleaning :URL Cleaning

```
df$ClickURLCleaned<-gsub(df$ClickURL,pattern = "http://|https://|www\\.",replacement = "")
```

Changing Class of datetime from factor to datetime

```
df$QueryTime<-as.POSIXct(df$QueryTime)
```

## Summary Statistics

Total no. of Queries

```
total_queries<-as.numeric(nrow(df))
total_queries
```

```
## [1] 3537372
```

Instances of New Queries / Unique queries in the log

```
new_queries <- as.numeric(length(unique(df$Query)))
new_queries
```

```
## [1] 1219923
```

Total Unique Users

```
unique_users<-length(unique(df$AnonID))
unique_users
```

```
## [1] 61592
```

Using simple calculations, I came to a conclusion a user fires at least 19 queries in the log

```
query_per_user <- new_queries/unique_users
query_per_user
```

```
## [1] 19.80652
```

**Calculating Next Page Requests** : As per AOL,if the user requested the next "page" or results for some query, this appears as a subsequent identical query with a later time stamp

```
query<-"SELECT AnonID,Query,COUNT(*)as Count FROM df GROUP BY AnonID,Query"
next_page_req<-as.data.table(sqldf(query))
next_page_req<-next_page_req[next_page_req$Count>1,]
```

Looking at Next Page Requests

```
head(next_page_req,10)
```

```
##     AnonID                      Query Count
##  1:    479       allegory of the cave     3
##  2:    479             bose car decal     3
##  3:    479                 car decals     3
##  4:    479           citation machine     6
##  5:    479             cranial nerves     3
##  6:    479                 dictionary     3
##  7:    479 existence precedes essence     2
##  8:    479                eye muscles     2
##  9:    479         german translation     2
## 10:    479   kant theory of knowledge     6
```

```
#Number of Next Page Requests
nrow(next_page_req)
```

```
## [1] 589345
```

**Total Unique URLS Count**

```
unique_urls<-as.numeric(length(unique(df$ClickURLCleaned)))
unique_urls
```

```
## [1] 379146
```

**CLick-through events**: Those queries on which users has acted by clicking on one of the links in the results

```
click_through_events<-as.numeric(length(which(!is.na(df$ItemRank))))
click_through_events
```

```
## [1] 1902838
```

**Queries w/o click through:Those queries on which users has not taken any action**

```
without_click_through_events<-as.numeric(length(which(is.na(df$ItemRank))))
without_click_through_events
```

```
## [1] 1634534
```

# Detection of Missing Data

*Extracting date to an another column*

```
df$Date<-format(as.Date((df$QueryTime)), "%Y-%m-%d")
```

*Total days of data*

```
difftime(min(df$Date),max(df$Date),units = "days")
```

```
## Time difference of -92 days
```

Total log in days = 93 days including the first day

```
date_range<-seq(as.Date(min(df$Date)), by = "day",length.out = 93)
```

**Unique Dates in Log data**

```
unique_dates<- as.Date(unique(df$Date))
unique_dates
```

```
##  [1] "2006-03-01" "2006-03-02" "2006-03-03" "2006-03-04" "2006-03-06"
##  [6] "2006-03-07" "2006-03-08" "2006-03-09" "2006-03-13" "2006-03-14"
## [11] "2006-03-20" "2006-03-21" "2006-03-22" "2006-03-26" "2006-03-27"
## [16] "2006-03-30" "2006-03-31" "2006-04-06" "2006-04-11" "2006-04-18"
## [21] "2006-04-20" "2006-04-21" "2006-04-23" "2006-04-24" "2006-04-28"
## [26] "2006-05-01" "2006-05-04" "2006-05-05" "2006-05-08" "2006-05-09"
## [31] "2006-05-10" "2006-05-15" "2006-05-25" "2006-05-27" "2006-03-05"
## [36] "2006-03-10" "2006-03-17" "2006-03-23" "2006-03-24" "2006-03-28"
## [41] "2006-03-29" "2006-04-01" "2006-04-03" "2006-05-06" "2006-05-12"
## [46] "2006-05-13" "2006-05-30" "2006-03-15" "2006-03-16" "2006-04-04"
## [51] "2006-04-05" "2006-04-17" "2006-04-19" "2006-04-22" "2006-04-27"
## [56] "2006-05-18" "2006-05-22" "2006-05-31" "2006-03-25" "2006-05-11"
## [61] "2006-05-23" "2006-05-24" "2006-05-26" "2006-02-28" "2006-03-11"
## [66] "2006-03-12" "2006-04-07" "2006-05-03" "2006-03-18" "2006-03-19"
## [71] "2006-04-02" "2006-04-08" "2006-04-09" "2006-04-10" "2006-04-15"
## [76] "2006-04-16" "2006-04-25" "2006-04-26" "2006-04-29" "2006-04-30"
## [81] "2006-05-02" "2006-05-07" "2006-05-14" "2006-05-16" "2006-05-20"
## [86] "2006-05-21" "2006-05-29" "2006-04-14" "2006-05-19" "2006-05-28"
## [91] "2006-04-13" "2006-04-12" "2006-05-17"
```

**Results whether any data is missing from any date from start date of logs and end date of logs

```
missing_dates<-which(!(unique_dates %in% date_range))
missing_dates
```

```
## integer(0)
```

```
length(missing_dates)
```

```
## [1] 0
```

**Result:** This implies there is no missing search query logs for any date

**Queries Per Day**

```
query<-"SELECT Date,Count(*) as Queries_per_day FROM df GROUP BY Date ORDER BY Date"
query_day<-sqldf(query)
query_day$Date<-as.Date(query_day$Date)
```

*Summarizing Query per day*

```
summary(query_day)
```

```
##       Date            Queries_per_day
##  Min.   :2006-02-28   Min.   : 4330
##  1st Qu.:2006-03-23   1st Qu.:31985
##  Median :2006-04-15   Median :40506
##  Mean   :2006-04-15   Mean   :38036
##  3rd Qu.:2006-05-08   3rd Qu.:44772
##  Max.   :2006-05-31   Max.   :51990
```

*Minimum queries in a day*

```
min(query_day$Queries_per_day)
```

```
## [1] 4330
```

*Maximumn queries in a day*

```
max(query_day$Queries_per_day)
```

```
## [1] 51990
```

On an average 38036 queries are fired in a day

```
query_day
```

```
##          Date Queries_per_day
## 1  2006-02-28            4330
## 2  2006-03-01           45071
## 3  2006-03-02           46660
## 4  2006-03-03           43020
## 5  2006-03-04           45748
## 6  2006-03-05           51990
## 7  2006-03-06           47758
## 8  2006-03-07           46531
## 9  2006-03-08           44867
## 10 2006-03-09           46651
## 11 2006-03-10           41577
## 12 2006-03-11           44617
## 13 2006-03-12           49799
## 14 2006-03-13           47454
## 15 2006-03-14           45195
## 16 2006-03-15           46928
## 17 2006-03-16           45656
## 18 2006-03-17           40823
## 19 2006-03-18           44858
## 20 2006-03-19           49227
## 21 2006-03-20           46401
## 22 2006-03-21           47888
## 23 2006-03-22           48291
## 24 2006-03-23           44772
## 25 2006-03-24           41118
## 26 2006-03-25           41879
## 27 2006-03-26           47807
## 28 2006-03-27           46024
## 29 2006-03-28           45298
## 30 2006-03-29           42388
## 31 2006-03-30           40506
## 32 2006-03-31           37365
## 33 2006-04-01           39263
## 34 2006-04-02           40725
## 35 2006-04-03           43160
## 36 2006-04-04           31948
```

```
## 37 2006-04-05          31465
## 38 2006-04-06          30710
## 39 2006-04-07          28224
## 40 2006-04-08          31400
## 41 2006-04-09          31191
## 42 2006-04-10          31189
## 43 2006-04-11          31895
## 44 2006-04-12          23628
## 45 2006-04-13          21745
## 46 2006-04-14          28977
## 47 2006-04-15          27618
## 48 2006-04-16          29412
## 49 2006-04-17          33044
## 50 2006-04-18          31363
## 51 2006-04-19          31985
## 52 2006-04-20          31489
## 53 2006-04-21          27922
## 54 2006-04-22          29768
## 55 2006-04-23          32551
## 56 2006-04-24          33099
## 57 2006-04-25          32267
## 58 2006-04-26          31070
## 59 2006-04-27          29163
## 60 2006-04-28          26551
## 61 2006-04-29          28975
## 62 2006-04-30          32440
## 63 2006-05-01          32115
## 64 2006-05-02          32252
## 65 2006-05-03          38218
## 66 2006-05-04          39281
## 67 2006-05-05          35949
## 68 2006-05-06          38342
## 69 2006-05-07          42534
## 70 2006-05-08          42140
## 71 2006-05-09          41565
## 72 2006-05-10          40913
## 73 2006-05-11          42757
## 74 2006-05-12          36700
## 75 2006-05-13          38536
## 76 2006-05-14          40807
## 77 2006-05-15          42352
## 78 2006-05-16          32014
## 79 2006-05-17           4411
## 80 2006-05-18          41608
## 81 2006-05-19          38382
## 82 2006-05-20          39331
## 83 2006-05-21          45293
## 84 2006-05-22          46279
## 85 2006-05-23          43304
## 86 2006-05-24          41132
## 87 2006-05-25          40465
## 88 2006-05-26          36809
## 89 2006-05-27          36290
## 90 2006-05-28          36133
```

```
## 91 2006-05-29           42765
## 92 2006-05-30           44291
## 93 2006-05-31           41670
```

We observe that on 2006-02-28 & 2006-05-17 only 4330 & 4411 queries were fired respectively on search engine which indicates may be some technical reason that data is not logged which is unusual as per usual behavior of users or may be servers were down which resulted in not logging of data while users were browsing

## Sessionizing the Query Log data

### Time-oriented approaches

Defining **Session** : more than 30 minutes between events is a new session

```
df$qt<-as.POSIXct(df$QueryTime)
df<-df[, session_id:=paste(cumsum((c(0, diff(qt))/60 > 30)*1)), by=AnonID]
df$session_id<-as.numeric(df$session_id)
total_sessions<-sum(df$session_id)
#Total no. of sessions in 93 days time period
total_sessions
```

```
## [1] 116024905
```

On an average, each session last for at least 32.8 mins

```
summary(df$session_id)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0     6.0    18.0    32.8    43.0   547.0
```

## *Summarizing Queries requested at Period of day*

Segmenting hours of day into different periods 0-7 <- Night 7-10 <- morning 10-12 <- Noon 12-17 <- Afternoon 17-23 <- Evening

```
hour<-as.numeric(format(df$QueryTime, '%H'))
df <- data.table(df,hour=hour,period=cut(hour, c(-Inf, 7, 10, 12, 17, Inf),
      labels=c("night", "morning", "noon", "afternoon", "evening")))
```

## Detection of Outliers

- Outliers and robots sessions were removed before analysis
- Outliers are long term user sessions containing too many queries which were probably generated by robots .So, Removing user sessions with highest no. of queries (top~1000)

```
query<-"SELECT AnonID,session_id,COUNT(*)as Count FROM df
GROUP BY AnonID,session_id ORDER BY Count DESC"
potential_outliers<-as.data.table(sqldf(query))
outliers<-head(potential_outliers,1000)
```

Number of Outliers we are going to remove

```
length(unique(outliers$AnonID))
```

```
## [1] 638
```

```
out_id<-unique(outliers$AnonID)
```

361455 such records are robots or identified as outliers

```
length(which(df$AnonID %in% out_id))
```

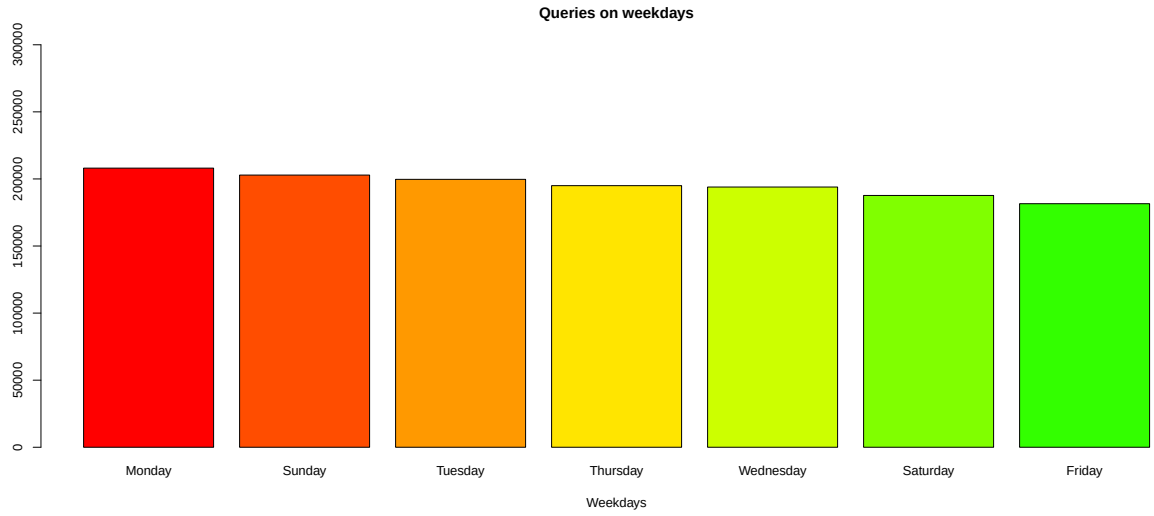
```
## [1] 361445
```

```
pos<-which(df$AnonID %in% out_id)

#Removing outliers from data frame
df<-df[-pos]
```

```
#Day wise Queries Trend
df$Day <- weekdays(as.Date(df$Date))
query<-"SELECT Day,Count(DISTINCT(Query)) as queries_count FROM df
GROUP BY Day ORDER BY queries_count DESC"
query_weekday<-sqldf(query)
```

```
barplot(query_weekday$queries_count,col=rainbow(20),
main = "Queries on weekdays",xlab = "Weekdays"
,names.arg = query_weekday$Day,
ylim =range(pretty(c(0,300000))))
```




On Monday and Sunday there are most queries fired. and Friday and Saturday there are less queries fired being the weekend days

*Hour wise Queries Trend*

```
query<-"SELECT Hour,Count(DISTINCT(Query)) as queries_count FROM df
GROUP BY Hour ORDER BY queries_count DESC"
query_hour<-sqldf(query)
query_hour$queries_count<-as.numeric(query_hour$queries_count)
```

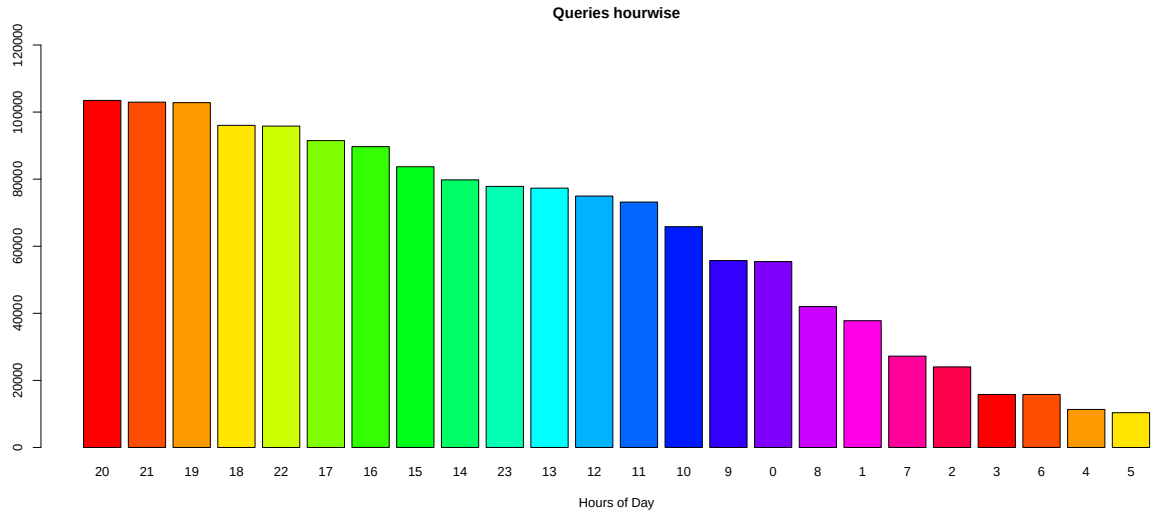
```
getOption("scipen")
```

```
## [1] 0
```

```
opt <- options("scipen" = 20)
getOption("scipen")
```

```
## [1] 20
```

```
barplot(query_hour$queries_count,col=rainbow(20),main = "Queries hourwise",
xlab = "Hours of Day",names.arg = query_hour$hour,ylim = range(pretty(c(11000,110000))))
```




We observe that most of the queries were fired around 8,7 & 10 in evening and then in afternoon and very less users in morning time which is usual trend.

*Session Analysis*: Session is defined as a sequence of consecutive queries submitted by a same user in sufficiently small time period(say 30 mins)

*Query Period*

```
query<-"SELECT period,Count(DISTINCT(AnonID)) as unique_users,
SUM(session_id) as total_sessions,Count(*) as queries_count FROM df
GROUP BY period ORDER BY queries_count DESC"
query_period<-sqldf(query)
query_period$per_users<-(query_period$unique_users/unique_users)*100

# 81% of users are active in evening and about 40% of users are active
#in night and morning.
query_period
```

```
##      period unique_users total_sessions queries_count per_users
## 1   evening        49705       35820639       1243663  80.70042
## 2 afternoon        45808       26618954        876703  74.37330
## 3     night        28519       13185245        416302  46.30309
## 4   morning        28580       10812163        336104  46.40213
## 5      noon        30339        9432474        303155  49.25802
```

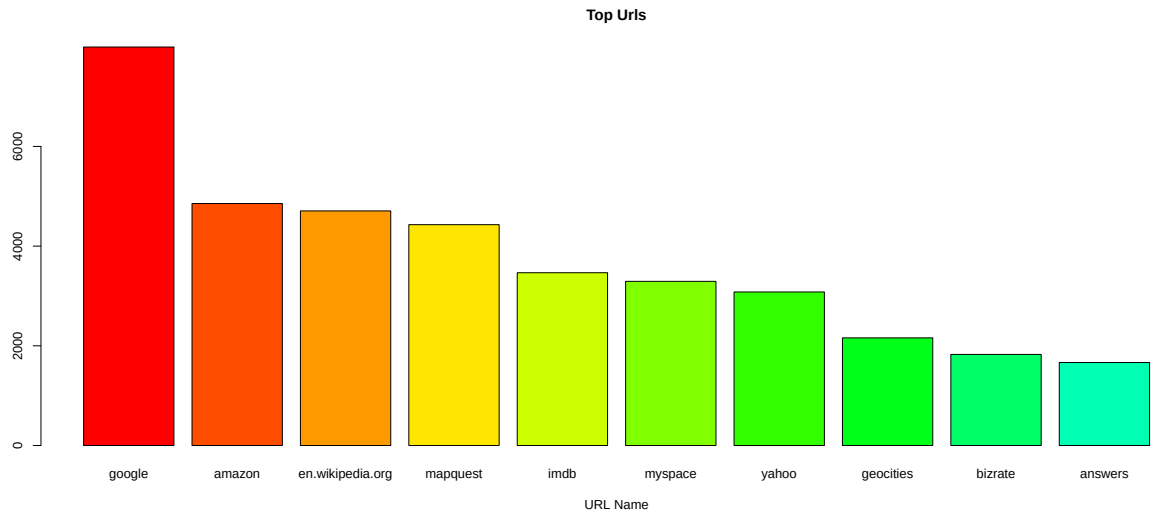
```
query<-"SELECT session_id,Count(DISTINCT(Query)) as query_count FROM df
GROUP BY session_id"
session_query<-sqldf(query)
```

Long sessions are likely during evening and afternonn and long breaks are more likely during night,noon and morning.

- Session length is seen as a more accurate alternative to measuring *pageviews*
- Sessions per user can be used as a measurement of **website usage**

```
sqlquery<-"SELECT ClickURLCleaned,COUNT(DISTINCT(AnonID)) as users_count FROM df
GROUP BY ClickURLCleaned  ORDER BY users_count DESC"
top_urls<-as.data.table(sqldf(sqlquery))
top_urls<-top_urls[-1,]
top_urls_plot<-head(top_urls,10)
top_urls_plot$ClickURLCleaned<-gsub(top_urls_plot$ClickURLCleaned,
                                    pattern = ".com",replacement = "")
```

```
barplot(top_urls_plot$users_count,col=rainbow(20),main = "Top Urls"
        ,xlab = "URL Name",names.arg = top_urls_plot$ClickURLCleaned)
```

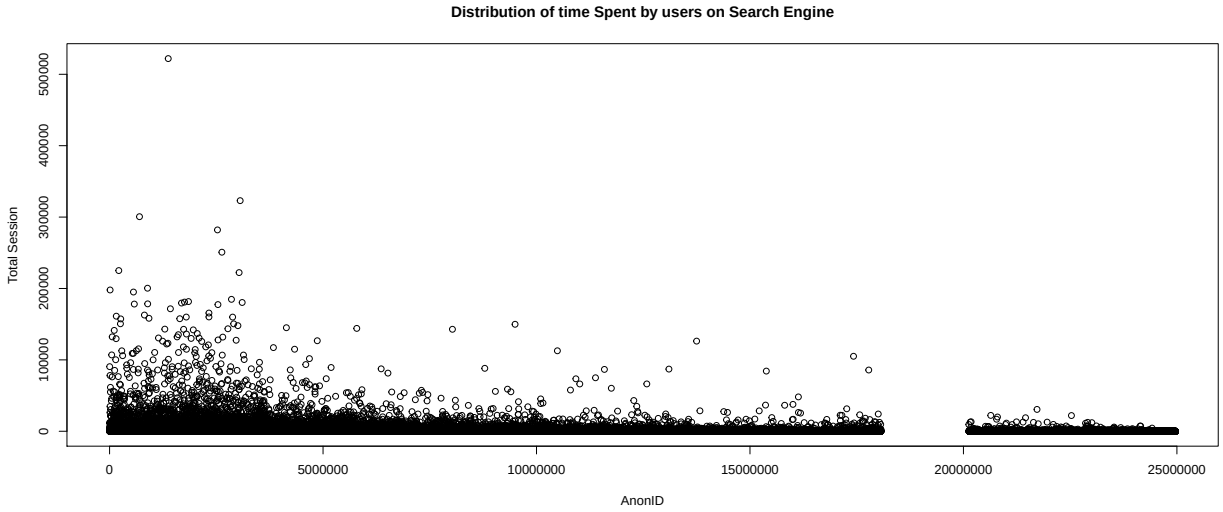



## ** Typical Time Spent by users on Search Engine **

```
sqlquery<-"SELECT AnonID,SUM(session_id) as total_sessions FROM df
GROUP BY AnonID"
time_spent<-sqldf(sqlquery)
#On an average each user spent 1573 session where each session is of 30 mins
#on search engine AOL in 93 days time
summary(time_spent)
```

```
##      AnonID          total_sessions
##  Min.   :     479   Min.   :     0
##  1st Qu.: 5847020   1st Qu.:     3
##  Median :11351943   Median :    57
##  Mean   :11813778   Mean   :  1573
##  3rd Qu.:16827847   3rd Qu.:   460
##  Max.   :24969423   Max.   :521993
```

```
#Disribution of Time Spent by users on search engine
plot(x=time_spent$AnonID,y=time_spent$total_sessions,
xlab = "AnonID",ylab = "Total Session",
main = "Distribution of time Spent by users on Search Engine")
```




## Analysing Queries that do not typically lead to click

Collecting users who were *active* on most of the days

```
sqlquery<-"SELECT AnonID,Count(DISTINCT(Date)) as date_count
FROM df GROUP BY AnonID ORDER BY date_count DESC"
active_users<-as.data.table(sqldf(sqlquery))
pos<-which(active_users$date_count>=50)
active_users_anonid<-active_users$AnonID[pos]

dt<-df[which(df$AnonID %in% active_users_anonid),]
dt_click<-dt[!which(is.na(dt$ItemRank))]
```

```
dt_noclick<-dt[which(is.na(dt$ItemRank))]

dt_noclick$totalwords <- sapply(dt_noclick$Query,
function(x) length(unlist(strsplit(as.character(x), "\\W+"))))

#On an average 3 words per query do not lead to click
summary(dt_noclick$totalwords)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    2.00    2.00    2.84    3.00   97.00
```

```
dt_click<-dt[!which(is.na(dt$ItemRank))]

dt_click$totalwords <- sapply(dt_click$Query,
function(x) length(unlist(strsplit(as.character(x), "\\W+"))))

#We  can conclude in afternoon and evening , there are more clicks
summary(dt_click$period)
```

```
##     night   morning      noon afternoon   evening
##     65777     58255     47623    126572    167795
```

```
#We can come to the conclusion that if we keep number of words short in query
#that has a probable chance of converting to a click.

dt_noclick$Query<-as.character(dt_noclick$Query)


#No click words Analysis
#I looked at finding the words with their corresponding frequency and then
#checked those words whether they are misspelled or not
words_noclick<- strsplit(dt_noclick$Query, "\\W")
words_no_click<-unlist(words_noclick)
freq<-table(words_no_click)
freq1<-sort(freq, decreasing=TRUE)
temp_table<-data.table(paste(names(freq1), freq1, sep=","))
words_noclick<-data.table(do.call(rbind, strsplit(temp_table$V1, ',')))
colnames(words_noclick)<-c("words","freq")
#Using qdap package dictionary to find out whether a word is misspelled or not
n_misspelled <- sapply(words_noclick$words, function(x){
  length(which_misspelled(x, suggest = FALSE))
})
miss_spelled<-data.frame(words_noclick$words,words_noclick$freq,
                         n_misspelled, row.names = NULL)
miss_spelled<-data.table(miss_spelled)
colnames(miss_spelled)<-c("word","freq","is_misspell")
miss_spelled$word<-as.character(miss_spelled$word)
miss_spelled$charlen<-nchar(miss_spelled$word)
#Removing words of 1 length
miss_spelled<-miss_spelled[-which(miss_spelled$charlen==1)]
remove_words<-c("com","of","www","in","the","for","and","to","http",
                "is","you","org","on","how","at","org")
```

```
stop_words<-stopwords("english")
#Removing functional words and stopwords
miss_spelled<-miss_spelled[-which(miss_spelled$word %in% remove_words)]
miss_spelled<-miss_spelled[-which(miss_spelled$word %in% stop_words)]

miss_spelled_words_len<-sum(miss_spelled$is_misspell)
total_words<-nrow(miss_spelled)

head(miss_spelled,20)
```

```
##                 word  freq is_misspell charlen
##  1:                  17177           0       0
##  2:              ebay  2675           1       4
##  3:               new  2370           0       3
##  4:              free  2150           0       4
##  5:               net  1976           0       3
##  6:              pogo  1651           1       4
##  7:             yahoo  1509           0       5
##  8:            lyrics  1339           0       6
##  9:            county  1306           0       6
## 10:           myspace  1267           1       7
## 11:          american  1256           0       8
## 12:            school  1255           0       6
## 13:            google  1210           0       6
## 14:              york  1132           0       4
## 15:              home  1083           0       4
## 16:              city  1054           0       4
## 17:          pictures  1019           0       8
## 18:              sale   929           0       4
## 19: letstalkhonestly   888           1      16
## 20:               aol   873           1       3
```

- More than 72 percent of the words are incorrectly spelled that leads to no click
- This implies product manager needs to deploy a spell correction feature on AOL for more clicks

```
per_miss_spelled_words_noclick<-miss_spelled_words_len/total_words
per_miss_spelled_words_noclick
```

```
## [1] 0.7236008
```

### Queries that almost always lead to click

```
df_click<-df[!which(is.na(df$ItemRank))]
query<-"SELECT Query,Count(ItemRank) as clicks FROM dt_click
GROUP BY Query ORDER BY clicks DESC"
query_clicks<-sqldf(query)
query_clicks$Query<-gsub(query_clicks$Query,pattern = "www.|.com",replacement = "")
trim <- function (x) gsub("^\\s+|\\s+$", "", x)
query_clicks$Query<-trim(query_clicks$Query)
query_clicks<-query_clicks[-which(query_clicks=="-"),]
```

```
query_clicks<-query_clicks[-which(query_clicks==""),]
head(query_clicks,25)
```

```
##                    Query clicks
## 1                 google   5250
## 3                  yahoo   4362
## 4                    ebay   3084
## 5                 myspace   2156
## 6               mapquest   1099
## 7                  yahoo   1035
## 8               my space    758
## 9              freeslots    732
## 10                 yahoo    686
## 11                google    650
## 12               myspace    623
## 13            craigslist    591
## 14       bank of america    549
## 15            ask jeeves    531
## 16                   xxx    433
## 17 craigslists new york    416
## 18               hotmail    400
## 19                israel    395
## 20            yahoo mail    383
## 21                  iwon    366
## 22     letstalkhonestly    363
## 23             free porn    362
## 24               hotmail    360
## 25          ohio lottery    355
## 26       onlinebootycall    339
```

```
#Queries having average query length of 3 leads to a click
summary(query_clicks)
```

```
##     Query               clicks
##  Length:152084      Min.   :   1.000
##  Class :character   1st Qu.:   1.000
##  Mode  :character   Median :   1.000
##                     Mean   :   3.035
##                     3rd Qu.:   3.000
##                     Max.   :5250.000
```

### Types of Queries made by Active users at Night

```
query<-"SELECT period,Query,Count(*) as count FROM dt_click
GROUP BY Query ORDER BY count DESC"
query_user<-sqldf(query)
query_night<-query_user[which(query_user$period=="night"),]
head(query_night,20)
```

```
##     period               Query count
```

```
## 1    night              google  5250
## 8    night           my space   758
## 18   night             hotmail   400
## 26   night onlinebootycall.com   339
## 33   night          excite.com   269
## 35   night   coffeebreakarcade   263
## 38   night           map quest   250
## 39   night                wamu   249
## 44   night             manhunt   229
## 48   night            eros.com   216
## 52   night       yahoo finance   211
## 55   night          dictionary   206
## 61   night             meoland   195
## 71   night        kcci weather   177
## 73   night                 wwe   175
## 77   night      teen pussy org   163
## 85   night       my space .com   151
## 91   night        trip advisor   145
## 94   night     myspace layouts   144
## 100  night          ebaymotors   140
```

- More Porn searches were seen at night

### Types of Queries made by Active users around Afternoon

```
query_aft<-query_user[which(query_user$period=="afternoon"),]
head(query_aft,20)
```

```
##       period                    Query count
## 4  afternoon                     ebay  3084
## 5  afternoon                  myspace  2156
## 11 afternoon               google.com   650
## 14 afternoon          bank of america   549
## 17 afternoon     craigslists new york   416
## 21 afternoon                 iwon.com   366
## 22 afternoon www.letstalkhonestly.com   363
## 24 afternoon              hotmail.com   360
## 28 afternoon             fidelity.com   301
## 29 afternoon                   amazon   300
## 43 afternoon                  walmart   230
## 45 afternoon  george mason university   227
## 46 afternoon               rune scape   226
## 53 afternoon              www.msn.com   207
## 58 afternoon                   copart   198
## 63 afternoon         granny blow jobs   191
## 67 afternoon             ticketmaster   189
## 69 afternoon                craiglist   181
## 74 afternoon            jerusalempost   174
## 75 afternoon                   drudge   168
```

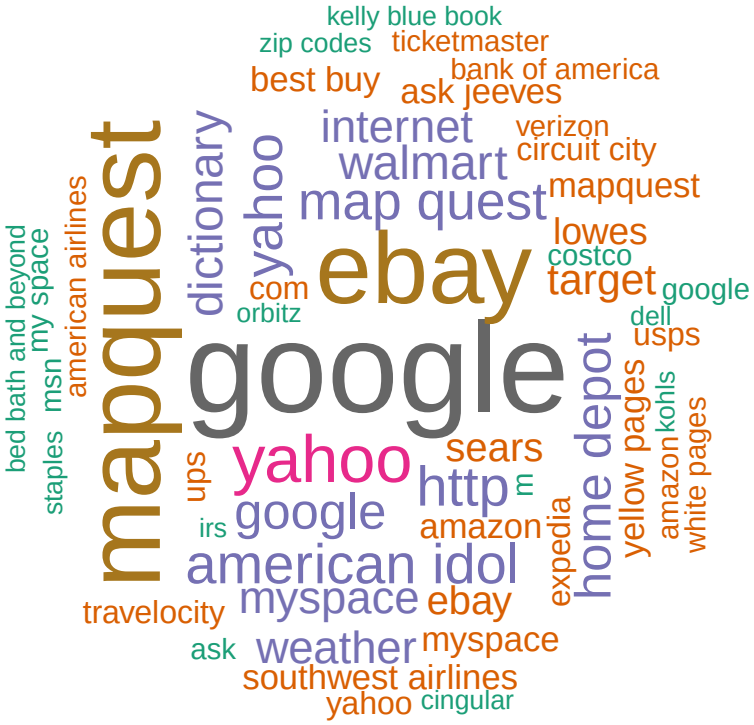
- It shows more of a shopping trend of users in afternoon going to sites like ebay,walmart,amazon

### Common Queries of Active users

```
sqlquery<-"SELECT Query,Count(DISTINCT(AnonID)) as users_count FROM dt
GROUP BY Query ORDER BY users_count DESC"
common_query<-as.data.table(sqldf(sqlquery))
common_query<-common_query[-1,]
common_query$Query<-gsub(common_query$Query,pattern = "www.|.com",replacement = "")
trim <- function (x) gsub("^\\s+|\\s+$", "", x)
common_query$Query<-trim(common_query$Query)
common_query<-common_query[-which(common_query$Query=="")]
#Some Common Queries Below with the users counts against them
head(common_query,25)
```

```
##                 Query users_count
##  1:             google         510
##  2:           mapquest         413
##  3:               ebay         386
##  4:              yahoo         226
##  5:               http         179
##  6:      american idol         178
##  7:          map quest         176
##  8:              yahoo         175
##  9:             google         156
## 10:         home depot         153
## 11:            walmart         144
## 12:         dictionary         142
## 13:            myspace         139
## 14:           internet         135
## 15:            weather         135
## 16:             target         127
## 17:               ebay         108
## 18:              sears         108
## 19:              lowes         100
## 20:            myspace          90
## 21:         ask jeeves          89
## 22:           best buy          88
## 23: southwest airlines          85
## 24:           mapquest          83
## 25:       yellow pages          82
##                 Query users_count
```

```
wordcloud(words = common_query$Query, freq = common_query$users_count, min.freq = 50,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```




## Relevance of Search Queries

- Queries that do not seem to have relevant results must be having an higher item rank since users has to navigate to next page, which results in increase in item rank
- Maximum Item Rank

```
rel<-df[which(!is.na(df$ItemRank)),]
dim(rel)
```

```
## [1] 1716246      12
```

- Maximum Item Rank is 500 which implies user browsed 500 next pages against a query

```
max(rel$ItemRank)
```

```
## [1] 500
```

```
sqlquery<-"SELECT Query,ItemRank,Count(*) as count FROM rel GROUP BY Query
ORDER BY ItemRank DESC"
no_rel_results<-as.data.table(sqldf(sqlquery))
```

## Top queries that do not seem to have relevant results along with Item Rank

```
head(no_rel_results,20)
```

```
##                                 Query ItemRank count
##  1:                  bang my husband      500     2
##  2:              john martin del campo      500     7
##  3:       newjersey mobilehome dealers      500     8
##  4:              photos young gay boys      500    30
##  5:                west bountiful utah      500    24
##  6:                      ralph manning      498    12
##  7:                             acuity      497    10
##  8:     ankle bracelet sexual meanings      497     3
##  9:                  judith a coulombe      497    14
## 10:               web cams that are on      495     4
## 11:                            vitello      488     6
## 12:                      arcadia press      483     3
## 13:           harley davidson t-shirts      483    19
## 14:       primitive style willow trees      476    37
## 15: mail order adult sex toy cataloges      475     7
## 16:                    porsche exhaust      475    89
## 17:          pharmacy tech final exams      471    13
## 18:                covered wagon lamps      470    29
## 19:         black panther collectables      465    15
## 20:                            bybee      464    26
```

## Relevance of Queries can be measured using following metrics

- *Stickiness* of users on websites will indicate relevance of search queries
- Session length is seen as a more accurate alternative to measuring page views
- Sessions per user can be used as a measurement of *website usage*

```
sqlquery<-"SELECT ClickURLCleaned,SUM(session_id) as total_sessions FROM df
GROUP BY ClickURLCleaned ORDER BY total_sessions DESC"
web_usage<-as.data.table(sqldf(sqlquery))
web_usage<-web_usage[-1,]
head(web_usage,20)
```

```
##         ClickURLCleaned total_sessions
##  1:          google.com         983436
##  2:           yahoo.com         739884
##  3:         myspace.com         612402
##  4:    en.wikipedia.org         409629
##  5:            ebay.com         408497
##  6:          amazon.com         367363
##  7:            imdb.com         319899
##  8:        mapquest.com         219252
##  9:      craigslist.org         188457
```

```
## 10:   bankofamerica.com         157810
## 11:      mail.yahoo.com         142878
## 12:             ask.com         115271
## 13:     tripadvisor.com         111597
## 14:       geocities.com         110762
## 15:         hotmail.com         109780
## 16:       freeslots.com         108008
## 17:         bizrate.com         100937
## 18:             msn.com          92617
## 19: profile.myspace.com          92021
## 20:          nextag.com          91272
```

This implies users browsing on AOL search engine look for other search engines and spend a lot of time on other search engines like google,yahoo and msn. So, AOL search engines is not performing well.

```
df_words<-df[-which(df$Query=="-")]
df_words<-df[!which(is.na(df$ItemRank)),]
df_words$totalwords <- sapply(df_words$Query,
function(x) length(unlist(strsplit(as.character(x), "\\W+"))))
sqlquery<-"SELECT totalwords,ItemRank,Count(*) as count
FROM df_words GROUP BY totalwords ORDER BY ItemRank "
words_rel<-as.data.table(sqldf(sqlquery))
head(words_rel,10)
```

```
##     totalwords ItemRank  count
##  1:          1        1 301338
##  2:          5        1 115834
##  3:         12        1   1047
##  4:         13        1    569
##  5:         17        1    152
##  6:         22        1     34
##  7:         30        1      3
##  8:         31        1      9
##  9:         33        1     12
## 10:         35        1      2
```

This implies a query of words having a count of 5 almost always convert to a click

## Insights that I would like to share with Product Manager are:

- People who come to browse on AOL search engines are looking for other search engines like google,yahoo & msn.
- spell correction feature should be added since 72% of words are incorrectly spelled by a users in queries that leads to a no click in collection of active users in 93 days.
- There are more Porn searches observed at night and shopping searches in afternoon.
- 81% of users are active in evening while 40% of users are active in morning and afternoon. So, evening is the best time to target users for ad.
- At 7,8 and 10 in the evening most queries are fired, so the best time to show sponsored links which would definitely results in conversion.
- Sunday and Monday are the best days in week to target users effectively.
- Looking at top urls we can say that users usually have an intent to ask questions when they come online since they are broesing in ask, answers.

- Product manager should approach amazon for campaigns since it is the second top url searched on search engine by users.
- Queries having a word count of less than or equal to 5 almost always converts to a click.