Twitter Data

Aaron Palumbo

10/4/2015

Dependencies

```
library(twitteR)
library(readr)
library(stringi)
library(R.utils)
```

```
## Loading required package: R.oo
## Loading required package: R.methodsS3
## R.methodsS3 v1.7.0 (2015-02-19) successfully loaded. See ?R.methodsS3 for help.
## R.oo v1.19.0 (2015-02-27) successfully loaded. See ?R.oo for help.
##
## Attaching package: 'R.oo'
##
## The following objects are masked from 'package:methods':
##
##     getClasses, getMethods
##
## The following objects are masked from 'package:base':
##
##     attach, detach, gc, load, save
##
## R.utils v2.1.0 (2015-05-27) successfully loaded. See ?R.utils for help.
##
## Attaching package: 'R.utils'
##
## The following object is masked from 'package:utils':
##
##     timestamp
##
## The following objects are masked from 'package:base':
##
##     cat, commandArgs, getOption, inherits, isOpen, parse, warnings
```

```
library(ggplot2)
```

Twitter Credentials

First we establish our twitter credentials

```
# remove carrige return
rmReturn <- function(s){
  gsub("\n", "", s)
}
```

```
consumer_key    <- rmReturn(read_file("consumer_key"))
consumer_secret <- rmReturn(read_file("consumer_secret"))
access_token    <- rmReturn(read_file("access_token"))
access_secret   <- rmReturn(read_file("access_secret"))

setup_twitter_oauth(consumer_key,
                    consumer_secret,
                    access_token,
                    access_secret)
```

```
## [1] "Using direct authentication"
```

Tweets

Now let's go ahead and pull down some tweets. We're looking for things with a #bigdata hash tag.

```
numTweets <- 500
tweets <- searchTwitter("#bigdata", n=numTweets)
```

Parse Tweets

Our goal is to try to look at the characteristics of the other hash tags people use with bigdata. We will now create a data frame from our tweets that we will subsequently push into Spark for analysis.

```
# function to extract id, username, hashtags, and timestamp from a tweet
parse_tweet <- function(tweet){
  id         <- tweet$id
  username   <- tweet$screenName
  time.stamp <- tweet$created
  content    <- tweet$text

  content <- gsub(pattern="[,\n]", replacement = " ", x = content)
  hashtags   <- unlist(
    lapply(unlist(strsplit(content, " ")),
           function(i){
             # check for # at beginning and
             # make sure we don't have any weird characters
             if (substr(i, 1, 1) == "#" & is.numeric(try(nchar(i)))){
               return(tolower(i))
             }
           })
    )
  df <- do.call(rbind,
                lapply(hashtags,
                       function(i) {return(data.frame(id=id,
                                                      user=username,
                                                      tag=i,
                                                      time=time.stamp))
                              }))
  return(df)
}

# apply that function to all our tweets
tdf <- do.call(rbind, lapply(tweets, parse_tweet))
# tdf
```

Process Tweets

Okay, now let's fire up Spark

```
library(SparkR)
```

```
##
## Attaching package: 'SparkR'
##
## The following objects are masked from 'package:R.oo':
##
##     clearCache, trim
##
## The following objects are masked from 'package:stats':
##
##     filter, na.omit
##
## The following objects are masked from 'package:base':
##
##     intersect, rbind, sample, subset, summary, table, transform
```

```
sc <- sparkR.init()
```

```
## Launching java with spark-submit command /home/apalumbo/workspace/cuny_msda_is622/spark-1.5.1-bin-ha
```

```
sqlContext <- sparkRSQL.init(sc)
```

Most popular hash tags

Now let's create a spark data frame from our local data frame

```
sdf <- createDataFrame(sqlContext, tdf)
```

We can now take a look at the ten most popular hashtags used with #bigdata

```
hashtag_counts <- summarize(groupBy(sdf, "tag"), count=n(sdf$tag))
head(arrange(hashtag_counts, desc(hashtag_counts$count)), 10)
```

```
##             tag count
## 1      #bigdata   448
## 2          #iot    76
## 3    #analytics    45
## 4  #datascience    43
## 5          #m2m    30
## 6          #api    25
## 7     #bigdata:    19
## 8        #cloud    18
## 9       #hadoop    16
## 10        #data    13
```

Estimating unique hash tags

Okay, we're going to take a stab at implementing the Flajolet-Martin algorithm to estimate the number of unique items.

First we need hash functions: (disclaimer: I have no idea how to create good hash functions. Hopefully this is sufficient)

```
# we will create a function to create a family of hash functions
create_hash <- function(){
  # Use random parameters to differentiate between family members
  a <- runif(n=1, min=1, max=100)
  b <- runif(n=1, min=1, max=100)
  f <- function(word){
    word <- as.character(word)
    n <- nchar(word)
    i <- 1:n
    hash <- (unlist(stri_enc_toutf32(word)) *  # foreach character create #
               a * i +                         # mult. according to place
               b * i)                          # add according to place
    hash <- intToBin(sum(hash))                # sum and convert to binary
    return(hash)
  }
  return(f)
}

trailingZeros <- function(hash){
  ss <- unlist(strsplit(hash, ""))
  length(ss) - max(which(ss == "1"))
}

processStream <- function(item){
  # expects two variables in external scope:
  # -- f.hash
  hashed <- unlist(lapply(f.hash, function(i) {i(item)}))
  unlist(lapply(hashed, trailingZeros))
}
```

Now we hash each tag with all our hash functions and keep track of the max trailing zeros (mtz). Our estimate of the number of unique elements is then 2^{mtz}.

We then break up our hash functions in to groups and take an average for each group. Finally, we make our final estimate by taking the median of the group means.

We will then compare that to the actual number.

```r
numHash <- 110
groupNum <- 11
f.hash <- lapply(1:numHash, function(i) {create_hash()})

estimates <- do.call(rbind, lapply(tdf$tag, processStream))
estimates.max <- apply(estimates, 2, max)
result <- median(
  apply(matrix(2**estimates.max, nrow=5), 2, mean)
  )
result
```

```
## [1] 579.2
```

```r
actual <- length(unique(tdf$tag))
actual
```

```
## [1] 271
```

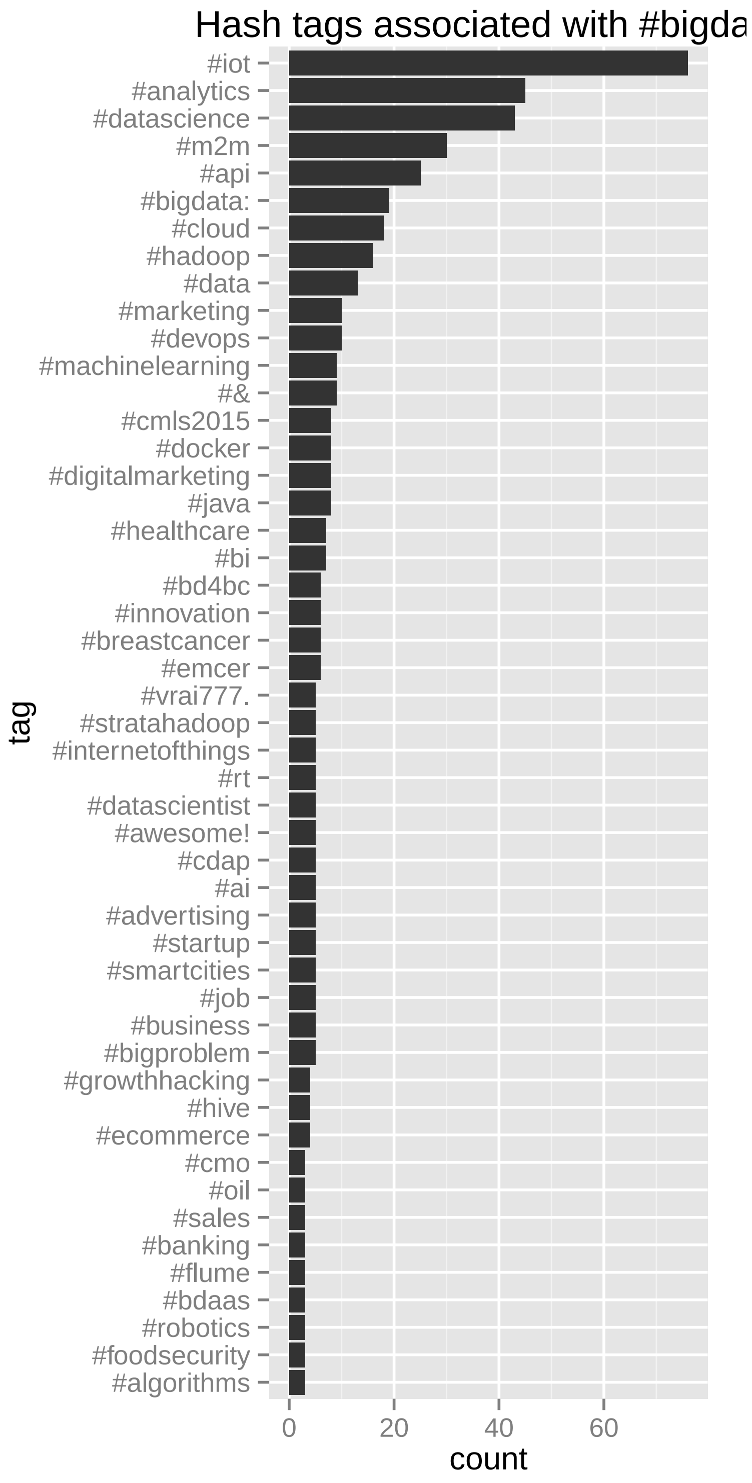
We see that our estimate of 579.2 isn't very close to the actual value of 271. Hopefully this is due to the small number of tweets we gathered (or maybe my hash functions).

Distribution of Hash Tags

Let's take a look at the top 50 hash tags. We already used spark to calculate this and the results are in our hashtag_counts.

```r
# convert to local data frame
df <- collect(hashtag_counts)
df <- df[order(df$count, decreasing = TRUE), ]
```

```r
numDisplay <- 50
df.display <- df[1:numDisplay, ]
df.display <- df.display[nrow(df.display):2, ]
df.display$tag <- factor(df.display$tag, levels=df.display$tag)
ggplot(data=df.display, aes(x=tag, y=count)) +
  geom_bar(stat="identity") +
  coord_flip() +
  ggtitle("Hash tags associated with #bigdata")
```



Not many surprises here, but still interesting. As noted above, we have almost 271 unique hash tags, so we don't expect to see a high concentration of single hash tags.

User distribution

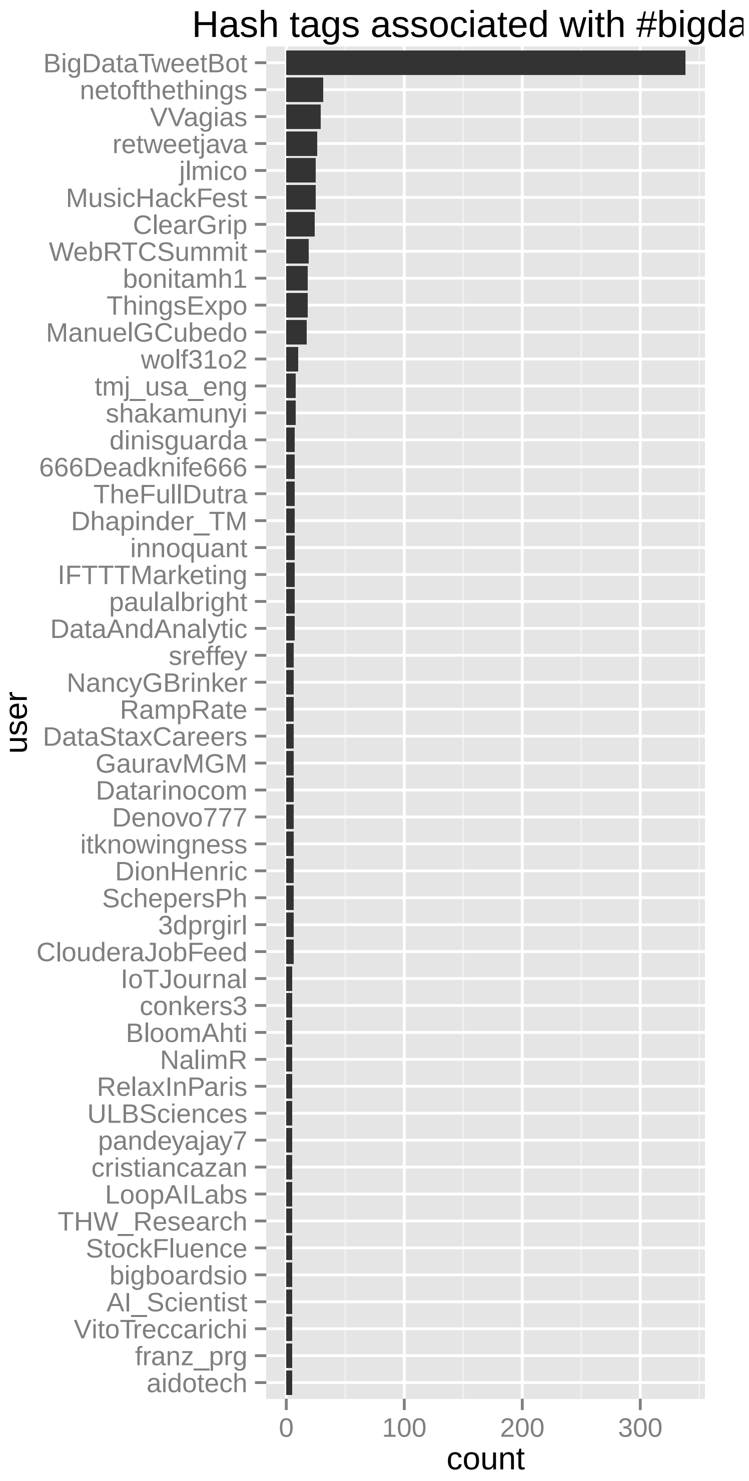
Let's see who the main people tweeting are:

```r
user_counts <- summarize(groupBy(sdf, "user"), count=n(sdf$user))
head(arrange(user_counts, desc(user_counts$count)), 10)
```

```
##               user count
## 1  BigDataTweetBot   338
## 2   netofthethings    31
## 3           VVagias    29
## 4      retweetjava    26
## 5           jlmico    25
## 6    MusicHackFest    25
## 7        ClearGrip    24
## 8     WebRTCSummit    19
## 9       ThingsExpo    18
## 10       bonitamh1    18
```

```r
# convert to local data frame
df <- collect(user_counts)
df <- df[order(df$count, decreasing = TRUE), ]
```

```r
numDisplay <- 50
df.display <- df[1:numDisplay, ]
df.display <- df.display[nrow(df.display):1, ]
df.display$user <- factor(df.display$user, levels=df.display$user)
ggplot(data=df.display, aes(x=user, y=count)) +
  geom_bar(stat="identity") +
  coord_flip() +
  ggtitle("Hash tags associated with #bigdata")
```



We can see from this chart that there are just a few users dominating the conversation. (Who is BigDataTweet-Bot?!).

BigDataTweetBot (@magicrat_larry): > I retweet #bigdata follow to get a feed of all that is tweeted about this subject. createdby @magicrat_larry