@Starbucks: Milestone Report

1. Introduction to the problem (based on your earlier Capstone submissions)

The focus of this project is to analyze its Starbucks' ***'content popularity'*** on Twitter. Twitter as a new form of social media can potentially contain much useful information. Because tweets are compact and fast, Twitter has become widely used to spread and share breaking news, personal updates and spontaneous ideas. Several recent studies examined Twitter from different perspectives, including the topological characteristics of Twitter, tweets as social sensors of real-time events, the forecast of box-office revenues for movies, etc. However, the explorations are still in an early stage and our understanding of Twitter, especially its large textual content, still remains limited. Due to the nature of microblogging, the large amount of text in Twitter, Starbucks may presumably contain useful information that can hardly be found in its traditional information sources such as the news media, distribution design, websites, company-owned retail stores or just plain word of mouth.

But this does not clearly suggest if the popularity for Starbucks on twitter is solely driven by its content. In order to properly evaluate this, the project is further developed on the theory that content is what drives SB's Twitter traffic. To test this hypothesis, we perform Topic Modelling and Linear Regression on a representative sample of SB's twitter data from July'16 and August'16. To begin with, we specifically try to answer the following questions:

- What are the types of topics covered on SB's twitter channel
- What are the common characteristics of these latent topics?
- Do they trigger an information spread (In this case- retweet count)?
- Do certain categories and types of topics trigger more information spread on Twitter?

2. A deeper dive into the data set:

The dataset is created using 'search' method of Twitter's API for R. The Twitter Search API is part of Twitter's REST API. It allows queries against the indices of recent or popular Tweets and behaves similarly to, but not exactly like the Search feature available in Twitter mobile or web clients. The Twitter Search API searches against a sampling of recent Tweets published in the past 7 days from when you run the query. The query operator used in this project is **'@starbucks'** and the tweets are extracted for the months of July & August as permitted by the API. The extraction is preceded by a mandatory user authentication process. To understand the process in detail, please refer to ***https://dev.twitter.com/oauth/application-only.*** To read about the API and understand the twitter authentication process in detail, please refer to ***https://dev.twitter.com/rest/public/search.***

3. *What important fields and information does the data set have?*

This process of querying is continued till the API rate limit is reached. For more understanding of the rate limit please refer to ***https://dev.twitter.com/rest/public/rate-limiting.*** We have two datasets, one for the month July (result_Max) and one for the month of August (result_max_new) and a total of 84090 observations

text	**The actual text of the status account '@starbucks'**
favorited	**Indicates whether this Tweet has been liked by the authenticating user.**
favoriteCount	**Indicates approximately how many times this Tweet has been "liked" by Twitter users.**
replyToSN	**Screen name of the user this is in reply to**
created	**When this status was created**
truncated	**Whether this status was truncated**
replyToSID	**If the represented Tweet is a reply, this field will contain the string representation of the original Tweet's ID.**
id	**The integer representation of the unique identifier for this Tweet.**
replyToUID	**If the represented Tweet is a reply, this field will contain the integer representation of the original Tweet's author ID.**
statusSource	**Source user agent for this tweet**
screenName	**Screen name of the user who posted this status**
retweetCount	**The number of times this status has been retweeted**
isRetweet	**Whether this tweet is a retweet of another tweet or status**
retweeted	**TRUE if this status has been retweeted**

longitude	**Represents the geographic location of this Tweet as reported by the user or client application**
latitude	**Represents the geographic location of this Tweet as reported by the user or client application**

Table 1: Unprocessed data attributes extracted using the Twitter API

4. *What are its limitations i.e. what are some questions that you cannot answer with this data set?*

This dataset has the following limitations:

- Text data is highly unstructured and does not always comply with the rules of the program used to identify and categorize words, phrases and clauses in a particular language. Idiomatic expressions, abbreviations, acronyms and business-specific lingo can all cause noisy text. It is particularly common with twitter data.
- This brings us to the second limitation. In the extracted data each retweet is identified as a separate text. This will have an adverse effect on our regression analysis that we will be performing shortly. There is a high possibility for our model to be inaccurate owing to the recurring content. To avoid this, we may have to remove all the duplicate text and this in turn may result in a poor regression model.
- In order to accurately determine the information spread, it is important for us to have additional attributes such as total followers/friends, geolocations for the different screen names in the dataset. In addition to the memory issues, we had challenges with the Twitter API's inconsistency, due to which we were unable to collect this more potentially useful information.

5. *What kind of cleaning and wrangling did you need to do?*

We now get a sense of what is contained in the data. So the first step is to import these texts into one's favorite computing environment, in our case R. Simultaneously it is important to organize and structure the texts to be able to access them in a uniform manner. We start structuring our data using regular expressions and remove some of the columns that are unnecessary.

```
# Combine both the datasets in to one and remove unnecessary columns from both the datasets
result_clean$favorited <- NULL
result_clean$favoriteCount <- NULL
result_clean$truncated <- NULL
result_clean$statusSource <- NULL
result_clean$retweeted <- NULL
result_clean$latitude <- NULL
result_clean$longitude <- NULL
result_clean$replyToSN <- NULL
result_clean$replyToSID <- NULL
```

```
result_clean$replyToUID <- NULL

# Arrange the data in descending order of the 'retweetCount' attribute
result_clean_sorted <- arrange(result_clean, desc(retweetCount))

# Convert the entire text to lower case
result_text_vector <- tolower(result_clean_sorted$text)
result_clean_sorted$text <- result_text_vector


# Remove old style retweets
result_text_vector = gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "", result_text_vector)

# Remove everything except english letters and space- symbols, numbers, punctuations, emoticons, etc
result_text_vector = gsub("@\\w+", "", result_text_vector)
result_text_vector = gsub("[[:punct:]]", "", result_text_vector)
result_text_vector = gsub("[[:digit:]]", "", result_text_vector)
result_text_vector = gsub("http\\w+ |^ http | http", "", result_text_vector)
result_text_vector = gsub("^rt | rt", "", result_text_vector)

# Save cleaned dataset and continue with topic modelling
write.csv(result_clean_sorted, "result_final_new.csv", row.names = F)
```

6. *Any preliminary exploration you've performed and your initial findings.*

For "classical" text mining tasks, we normally create so-called document-term matrix, probably the most common format to represent texts for computation. For this there is a need for a conceptual entity similar to a database holding and managing text documents in a generic way: we call this entity a text document collection or corpus. To use this, we install and load the 'tm' package. In addition to some text data preparation like removal of stop words, whitespaces and stemming, the 'tm' package also aids in some general visualization of data such as the most frequently occurring terms in the corpus. This is further complimented with a word cloud (implemented using the 'word cloud' package).

#freq.terms:

```
[1] "back"    "barista"  "brew"    "card"     "coconu
t"  "coffee"  "cold"    "cup"


[9] "drinks"   "free"    "going"   "great"     "ice"
"iced"    "macchiato" "milk"


[17] "mocha"    "morning"  "please" "really"   "star
bucks" "still"    "store"    "tea"


[25] "thank"    "time"     "try"     "work"
```

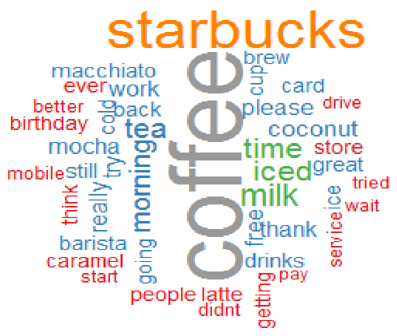

Fig 1: Word cloud to visualize the frequently occurring words in the dataset

A quick look at 'freq.terms' gives us an idea of the words that occur at least 500 times in the corpus. It is interesting to note that there aren't as many seasonal buzz -words (such as "berrysangria", "sangria") as one would expect from this outcome. Moving forward, using the Latent Dirichlet Allocation method, we generate six topics and five terms under each topic. Here's what the outcome looks like:

Topic 1	***Topic 2***	***Topic 3***	***Topic 4***	***Topic 5***	***Topic 6***
starbucks	*coffee*	*coffee*	*milk*	*coffee*	*coffee*
iced	*starbucks*	*starbucks*	*iced*	*free*	*starbucks*
coffee	*mocha*	*time*	*coconut*	*tea*	*morning*
card	*time*	*store*	*going*	*thank*	*milk*
caramel	*ever*	*work*	*macchiato*	*drinks*	*iced*

As observed in frequent terms and word cloud, there are more regular terms under each topic than certain buzz-words that we expect to identify. In order to accurately analyze the popularity quotient of each of these terms, it is important for us to first name each term and then, perform feature engineering on our dataset to create a relationship between the underlying features. Meaning, our dataset has 40,512 unique observations or texts and each text will be a mixture of each of the thirty terms. In order to accurately determine that, we add binary features for each term and insert those binary features as separate columns. We do that till we have 30 new attributes in our dataset. Additionally we parse the 'created' column to form 11 extra columns. This feature helps us to identify if there a particular day, or time in the day that the featured terms attract traffic.

```
# Add the 30 terms as 30 new columns in the dataset with each observation being a binary indicator of the presence of the term in the text
result_text_vector_new <- as.character(result_final_new$text)
result_final_new <- result_final_new %>% mutate(CaramelFix_1 = ifelse(grepl(pattern = "starbucks", x = result_text_vector_new), 1,0))
result_final_new <- result_final_new %>% mutate(CaramelFix_2 = ifelse(grepl(pattern = "iced", x = result_text_vector_new), 1,0))
result_final_new <- result_final_new %>% mutate(CaramelFix_3 = ifelse(grepl(pattern = "coffee", x = result_text_vector_new), 1,0))
result_final_new <- result_final_new %>% mutate(CaramelFix_4 = ifelse(grepl(pattern = "card", x = result_text_vector_new), 1,0))
result_final_new <- result_final_new %>% mutate(CaramelFix_5 = ifelse(grepl(pattern = "caramel", x = result_text_vector_new), 1,0))

# Include 'month' from the 'created' attribute as a new column
result_final_new$month=sapply(result_final_new$created, function(x) {p=as.POSIXlt(x);format(p, "%m")})

# Include 'day of the week' from the 'created' attribute as a new column
result_final_new$day=sapply(result_final_new$created, function(x) {p=as.POSIXlt(x);format(p, "%a")})

# Include 'date' from the 'created' attribute as a new column
result_final_new$date=sapply(result_final_new$created, function(x) {p=as.POSIXlt(x);format(p, "%d %b")})

# Include two separate columns to identify weekdays and weekends
day_vector <- result_final_new$day
result_final_new <- result_final_new %>% mutate(wday = ifelse(grepl(pattern = "Mon|Tue|Wed|Thu|Fri", x = day_vector), 1,0))
result_final_new <- result_final_new %>% mutate(wend = ifelse(grepl(pattern = "Sat|Sun", x = day_vector), 1,0))

# Include 'hour' from the 'created' attribute as a new column
result_final_new$hour=sapply(result_final_new$created, function(x) {p=as.POSIXlt(x);format(p, "%H")})

# Process the hour value to include distinct columns for daytime, evening and night
time <- as.numeric(sub("(\\d{1,2}):.*", "\\1", result_final_new$hour))
result_final_new$LateNight <- ifelse(00 <= time & time <= 04, 1,0)
result_final_new$Morning <- ifelse(04 < time & time <= 12, 1,0)
result_final_new$Afternoon <- ifelse(12 < time & time <= 16, 1,0)
result_final_new$Evening <- ifelse(16 < time & time <= 20, 1,0)
result_final_new$Night <- ifelse(20 < time & time <= 23, 1,0)
```

The resultant dataset has 47 new columns with 41 new features. Given the nature of our dataset, the scope for exploratory data analysis is limited but a quick EDA before proceeding further with linear regression and trees, shows the following results.

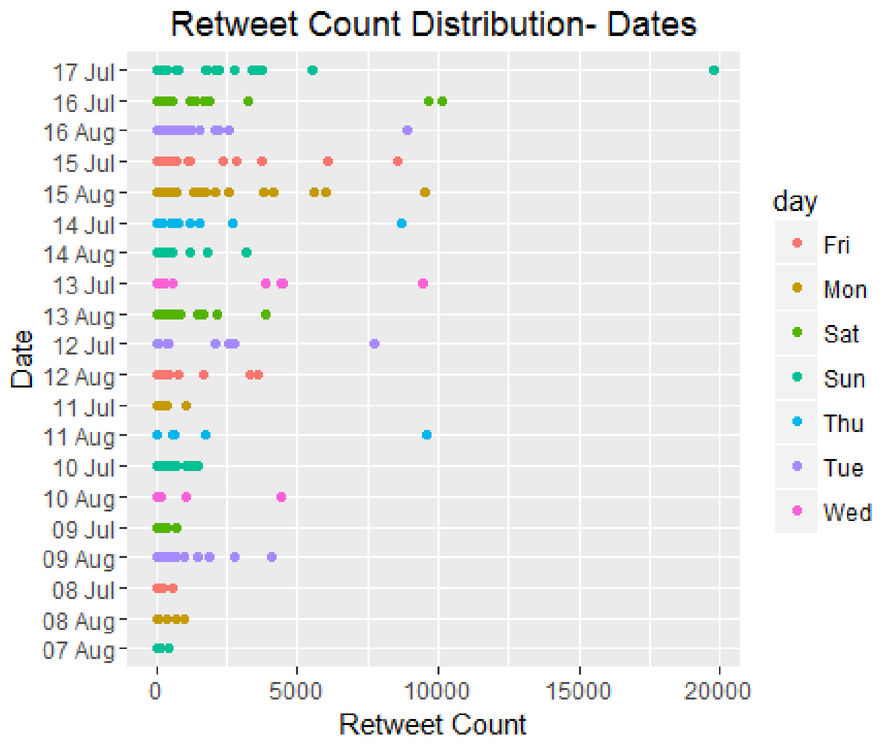

Fig 2: Distribution of Retweet Count over dates

```
# Retweet count distribution over 'dates' attribute
ggplot(result_final_new, aes(x = retweetCount, y = date))
+ geom_point(aes(col = day)) + xlab("Retweet Count") +
ylab("Date") + ggtitle("Retweet Count Distribution- Dates")
```

Fig 2 gives us an overview of the dates where the retweetCount is highest – 7/17/2016. A quick look at the data to identify the text, shows this:

"Ordered my drink @Starbucks Asked the barista if she wanted my name. She winked and said. "We gotcha" #JodieFoster"

This is a very interesting observation as it creates a possibility for us to develop this analysis further towards social network analysis. The plot is also successful is giving us the comparable data between the same dates in each month and it can be observed that a large number of tweets have a retweet count of 0 to less than 5000 followed by the range of 5000-10000.

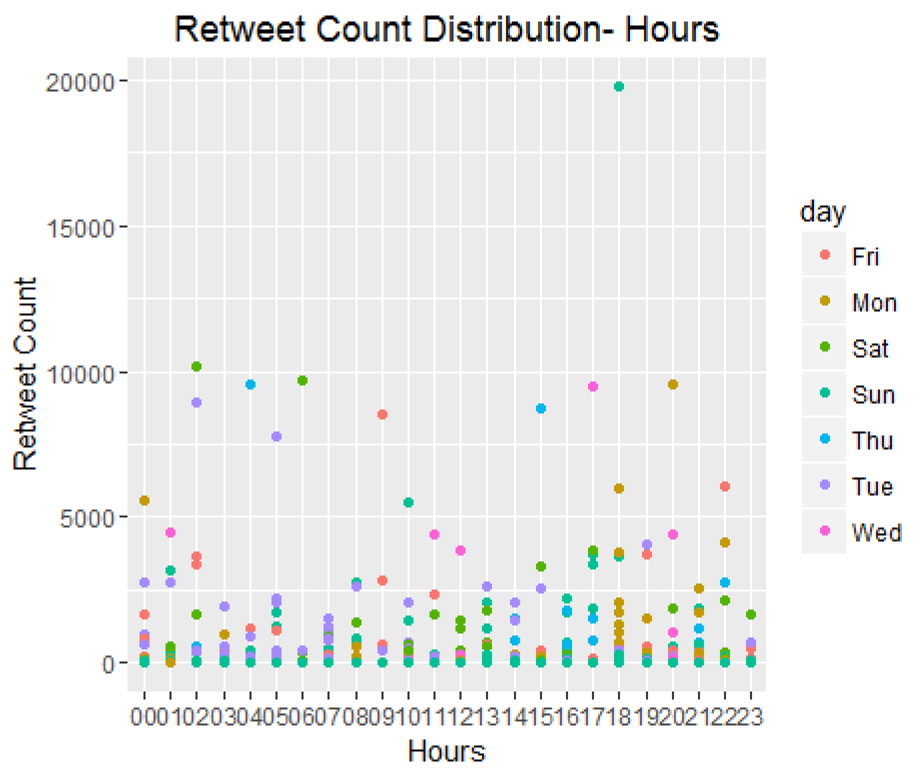

Fig 3: Distribution of Retweet Count over Hours

It is also important to know the time at which there was maximum information spread and for that, we create a similar plot using the 'hour' attribute. If we exclude the one outlier, we can notice that most of the information spread took place either in the ***Latenight-Earlymorning*** or **Evening** periods.

```
#Retweet count distribution over 'hour' attribute
ggplot(result_final_new, aes(x = hour, y =
retweetCount)) + geom_point(aes(col = day)) +
xlab("Hours") + ylab("Retweet Count") +
ggtitle("Retweet Count Distribution- Hours")
```

Another interesting visualization could be the information spread between weekdays and weekends.

```
# Retweet count distribution over 'day' attribute
ggplot(result_final_new, aes(x = day, y = retweetCount)) +
geom_point(aes(col = hour)) + facet_grid(wday ~.) +
geom_line(colour = "plum") + xlab("Days") + ylab("Retweet
Count") + theme(legend.position='none') + ggtitle("Retweet
Count Distribution- Days")
```

While the plot shows the numbers, we can a quick calculation to know the total number of retweet counts for weekdays and weekends to generally compare the two.

```
DT <- as.data.table(result_final_new)
DT[, list(totalretweetCount = sum(retweetCount), num = .N), by
= wday]
```

```
      wday    totalretweetCount   num
  1:   0           126427        11609
  2:   1           230985        28903
```

This means that we have 28903 weekday observations with a total retweet count of 230985 and 11609 weekend observations with a total of 126427. The data for wend is noticeably biased due to the one outlier with a retweet count of '19774'. Subtracting that from the original sum gives us '106653'

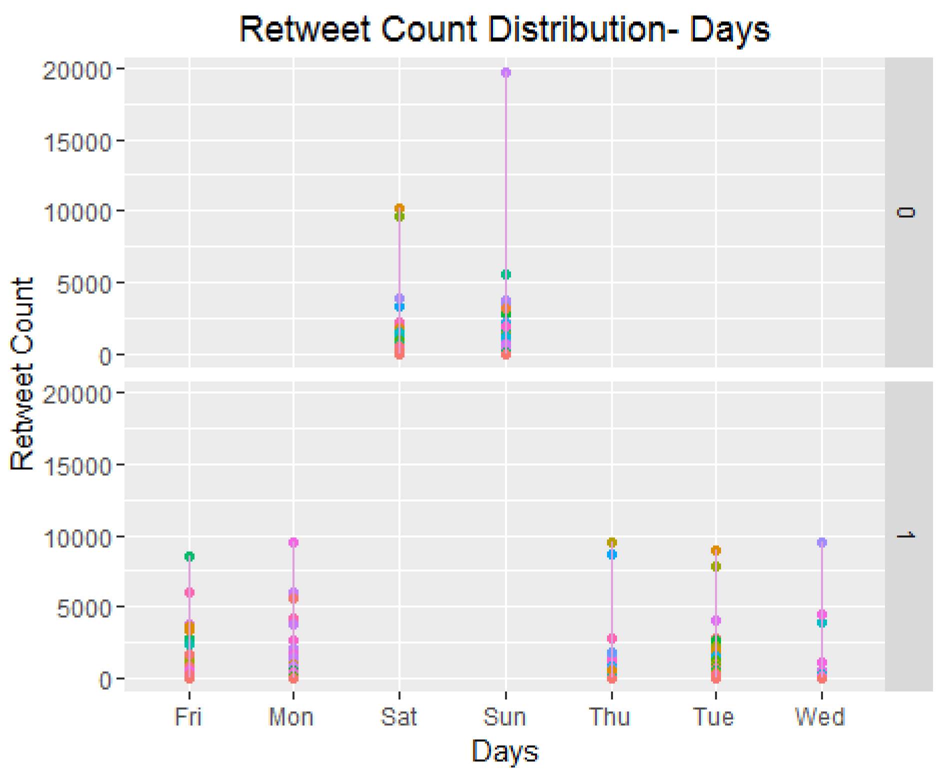

Fig 4: Distribution of Retweet Count: 'wend/wday' attribute

7. ***Based on these findings, what approach are you going to take? How has your approach changed from what you initially proposed, if applicable?***

In order to make sense of the LDA outcome for our dataset, we will run linear regression and decision trees to see if the content is indeed responsible for the information spread (retweet count in this case). If the outcome for the lm model is significantly high, we will prove our hypothesis right and vice-versa will open doors for further analysis such as social network analysis, news media analysis, analysis of Starbucks' social media strategies, etc.