Examining Broadcast Television Viewership Through Twitter

James Duvall : DC-DAT-9 : December 8, 2015

Public GitHub: https://github.com/jamduval/twitter_tv

1. Introduction

For some people, watching television in the evening is a cathartic way to de-stress from a hectic daily schedule. With such a wide range of programs to choose from, between reality programming such as *The Voice* or *Keeping Up With the Kardashians* and scripted programming such as *Arrested Development* or *Pushing Daisies*, there is something for everyone to enjoy. So it comes as no surprise when a show cancellation incites anger among viewers who had become invested in its characters. These confused, depressed viewers then seek consolation from their friends and family. Life now seems meaningless without their favorite space bounty hunter entertaining their lives once every week. Answers and reasons are fiercely sought after, and most of the time are attributed (in a major way) to the Nielsen Company, who publishes daily household ratings which shed light on the popularity of a show.

This project examines broadcast television with a focus on the "big four" networks (CBS, NBC, ABC, and FOX) and the CW. These stations offer the widest range of original scripted programming from Monday to Friday with CBS, NBC, and ABC airing between 8 p.m. and 11 p.m. EST, and FOX and the CW airing between 8 p.m. and 10 p.m. EST. Sunday is the exception where programming begins at 7 p.m. EST. More importantly, these stations are available to anyone with a TV (though some areas may require an antennae) and therefore derive their profits solely from the money spent by advertisers. As a result, the rating a program receives is vitally important to the life of a show.

At the crux of these ratings is the one rating to rule them all, the household adults 18–49. While other ratings may shed light on more accurate demographics of a program; for broadcast television, the household adults 18–49 live + same day rating is the nirvana all others aspire to reach. The household adults 18–49 live + same day (to 3 a.m. EST the following day through DVR viewership) rating represents the percentage of adults aged 18–49 who live in a household, in the U.S., and own a TV. This is the primary rating advertisers use to figure out how much they should be paying the networks to air their commercials, and will therefore make or break a broadcast program. While this may seem outdated, the alternative seems to be for the consumer to pay the network directly to get access to scripted shows, pushing live (sports/awards shows/etc...) and reality programming to be at the forefront of broadcast television in the coming years. Premium networks such as HBO and Showtime, and newer entrants such as Netflix and Amazon, don't require commercials to fund their shows and see most of their revenue from subscribers. Additionally, there seems to a quality push to these premium networks as they dominate awards season. The Nielsen household adults 18–49 live + same day rating (the "Nielsen rating") appears to be reaching the end of its life, but for now it still dominates the way advertisers think. In an attempt

to find a new method of measuring viewership, I used Twitter to capture the viewing behavior of the broadcast networks and compared the results to the Nielsen rating. The goal was that if Twitter could be used as a predictor for the Nielsen rating, then it could be used as an alternative source to measure a show's viewership.

2. Methodology

2.1. Twitter

Twitter is quickly becoming a source of infinite information as companies and bloggers attempt to figure out how to use its data to turn new insights into profits. Twitter has two application program interfaces (APIs) to allow users to obtain data: the REST API and the Streaming API. As my intent was to capture live viewership, the Streaming API (through a public stream) was most suitable to answer my question. The Streaming API simply spits out a stream of tweets. These tweets can be filtered by giving a list of text (official broadcast network program handles and hashtags, in my case) to subset the stream of incoming information. If I were to stream the data myself, it would require a persistent internet connection and access to a large amount of storage since hundreds of thousands of tweets, in JavaScript object notation (JSON) format, soon translate to terabytes of storage. To simplify the process, my instructor streamed a week's worth of data using the aforementioned list of text on which to filter the stream. Going forward, if I wanted to continue the work done in this project, then I would first need to obtain security keys from Twitter, then leverage Python to handle the incoming stream of data, and lastly use a storage platform, such as Amazon's Elastic Cloud Computing, which should allow for Twitter data to be streamed without requiring my computer to be constantly active.

The Twitter data I received was stored as a series of JSON files, where each tweet was its own JSON file and contained a large amount of user and text information. Below are most of the fields in a sample tweet and highlighted are the fields which were considered (and many went unused). While it would be nice if the below fields were perfect, this was not the case. Data was often missing, and some variables that seemed like great ways to filter the data, become cumbersome and nearly unusable (such as a user's location). One thing to note is that there was no real demographic data in the tweet, specifically age and gender. Other data scientists and researchers have devoted hours of time into producing papers and code to measure these two demographics, but for my purposes, I chose to ignore them. Please see *import_twitter_data.py* in my public GitHub for the Python program which read in the Twitter data from the list of JSON files and cleaned it into usable form.

Figure 1: Fields Available in a Tweet

There are three fields, however, that drove the basis of the data I could use:

(1) First and foremost, I added an indicator variable to indicate whether a show was "live + same day" or not. The `created_at` variable gives the coordinated universal time (UTC) of when the tweet was created. For simplification, I assigned a "1" to times between midnight and 11:05 a.m., and 0 otherwise. This translates to 7:00 p.m. to 6:05 a.m. Eastern Standard Time (EST), which is 11 hours long as a broad-stroke way to include both viewership data on the west coast and Sunday's programming beginning at 7:00 p.m. While the "live + same day" numbers from Nielsen stop at 3:00 a.m., I did not expect many non–pacific time tweets regarding television shows between 3:00 a.m. and 6:00 a.m., so I did not anticipate a strong bias based on this choice.

(2) There are two languages (`lang`) in the tweet itself: the user language and the tweet language. These two can differ, so while a user might indicate herself to speak French (the user portion), her tweet may be in English. I filtered this field based on the language of the tweet itself.

(3) There is also a very well populated `time_zone` field in the user portion of the tweet I required in order to try and keep only those tweets whose user resides in one of the four U.S. time zones.

2.2. Nielsen Data

For the other part of the data, I pulled in the Nielsen rating for each program across the "big four" broadcast networks (CBS, NBC, ABC, and FOX) and the CW. *TV by the Numbers* (TVbtN) is a web site devoted to blogging about television ratings and it was used to gather the Nielsen ratings. Using a combination of the Requests, Beautiful Soup, and the Pandas Python libraries, I grabbed a set of variables that TVbtN reports on their web site (please see Figure 2 below for how the result data looked). These included basic variables like the time the show aired and the name of the program (and the network) for

which I used to merge my datasets, the more important Nielsen rating, the share (which is a subset of the Nielsen rating calculated by eliminating households who have the TV turned off), and the total number of viewers (which represent people aged 2 and up). Note that the total number of viewers for broadcast television is essentially a useless statistic (though I still want to keep it for completeness). As an example, it can be dismaying when a show such as *NCIS* pulls in roughly 50 percent more total viewers than *Empire* but has a Nielsen rating of 50 percent less. This means that *NCIS* is an extremely older skewing program (or younger at less than 18, but I doubt it), and while there are more people watching it, the number of people that advertisers care to target is much less. Please see *TVBTN_web_scrape.py* for the program which pulled in this data, and then *twitter_main.py* for the program which ran a text file containing the URLs of each day's final Nielsen ratings through the web scraping program (and cleaned the data for analysis), both on my public GitHub.

Figure 2: Nielsen Data From TV by the Numbers

viewers	rating	share	day	air_date	air_time	show	network
17.97	2.3	8	Tue	2015-11-03	1900-01-01 20:00:00	NCIS	CBS
3.87	1.5	5	Tue	2015-11-03	1900-01-01 20:00:00	The Flash	The CW
4.56	1.4	5	Tue	2015-11-03	1900-01-01 20:00:00	The Muppets	ABC
5.06	1.4	4	Tue	2015-11-03	1900-01-01 20:00:00	Best Time Ever	NBC
3.18	1.0	3	Tue	2015-11-03	1900-01-01 20:00:00	Grandfathered	FOX
4.45	1.6	5	Tue	2015-11-03	1900-01-01 20:30:00	Fresh Off the Boat	ABC
2.52	0.8	3	Tue	2015-11-03	1900-01-01 20:30:00	The Grinder	FOX
8.22	2.0	6	Tue	2015-11-03	1900-01-01 21:00:00	The Voice	NBC
14.15	1.9	6	Tue	2015-11-03	1900-01-01 21:00:00	NCIS: New Orleans	CBS
3.84	1.4	4	Tue	2015-11-03	1900-01-01 21:00:00	Agents of SHIELD	ABC
2.44	1.0	3	Tue	2015-11-03	1900-01-01 21:00:00	Scream Queens	FOX
1.43	0.6	2	Tue	2015-11-03	1900-01-01 21:00:00	iZombie	The CW
8.11	1.8	6	Tue	2015-11-03	1900-01-01 22:00:00	Chicago Fire	NBC
7.86	1.5	5	Tue	2015-11-03	1900-01-01 22:00:00	Limitless	CBS
2.42	0.7	2	Tue	2015-11-03	1900-01-01 22:00:00	Wicked City	ABC

2.3. Problems With the Data

While trying to read in the data, I ran into a great deal of problems. The primary issue, however, was memory constraints. My computer, with a paltry 8gbs of RAM, could not read more than 300 thousand tweets into memory before stalling. I resolved this issue by sub-setting each day's worth of data into its own folder, and while this worked for the most part, it still required me to output each day's results to an excel file to be read in later for analysis through a separate Python script. If I could go back in time, I would have only looked at two programs from each network across the week to lessen the work my personal computer would have to handle. Additionally, some of the tweets contained "extra data" and could not be read in through the json.loads() process (maybe 20-30 in total), and so I used a simple try/except to skip them.

Another issue that arose while examining the data was that some tweets were clearly missing. As an example, I received essentially no tweets for *Empire.* This should not have been the case and is very clearly

an error; the source of which is unknown and was not isolated to *Empire*. One potential reason for this could have been the use of weekly or unofficial hashtags, which would have to be dealt with if I pursued this project in the long-term. Essentially, some networks push certain hashtags for the week. As an example, for the first few weeks of the new season of *Once Upon a Time*, #DarkSwan was trending for the show and some unique viewership might be captured by streaming that string. This would be dealt with by checking each show's official Twitter to see if they are pushing an episode-specific hashtag. However, this leads into the second issue, non–network specific hashtags. For example, #EmpireSeason2 is trending for *Empire*, but this is an unofficial reference not tied directly to the official twitter run by Fox. These sorts of issues would have to be dealt with on a week to week basis, likely by editing some script which alters the Twitter stream based on new trends in specific shows.

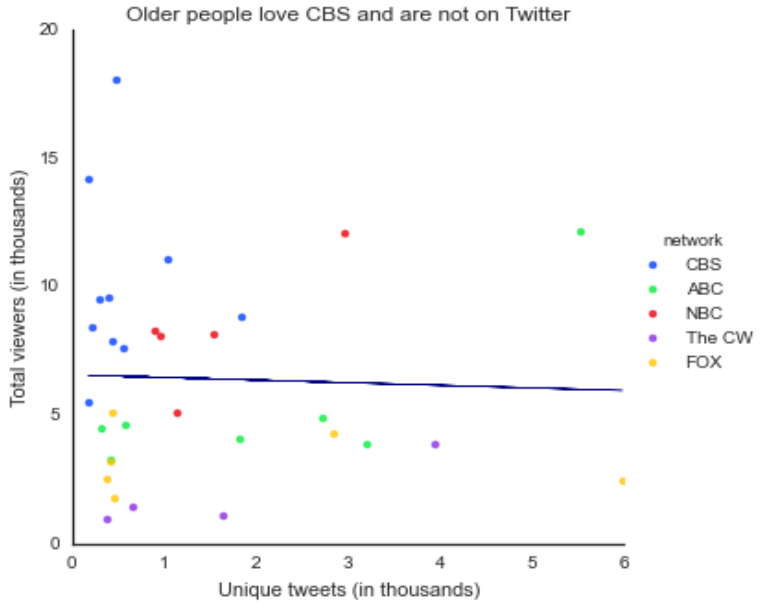
Lastly, I originally assumed that the Stream API would only print a tweet if it had a specific hashtag or handle. This was not the case as it also printed out any text with one of the hashtags or handles I suggested. For example, the word "limitless" is an exceptionally common word, which produces a wide range of tweets for which I did not want. In order to improve my analysis in the future, I would need to figure out a way around this to reduce the amount of information read in not only for the sake of my computer, but also to better manage the 200 tweet per second limit constraining the data retrieval process.

3. Results

3.1. Unique Tweets and Total Viewers

The first aspect of the data I want to display is the relationship between unique tweets and total viewers, primarily the negative coefficient (-0.10) on unique tweets in a simple OLS regression. This is expected in the sense that shows with more total viewers (i.e., specifically those with more viewers over the age of 49 such as *NCIS*) have less unique tweets because those older people are less likely to be on a younger-skewing social media platform like Twitter. Please see Figure 3 below illustrating this relatively weak but still useful relationship.

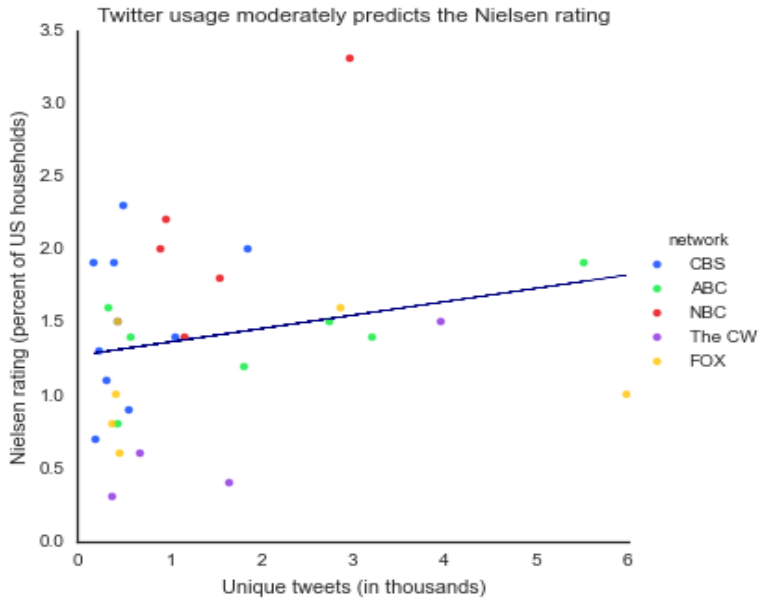
Figure 3: The Relationship Between the Number of Unique Tweets and Total Viewers



3.2. Unique Tweets and the Nielsen Rating

At the beginning of the project I asked the question whether the Nielsen rating could be predicted from the number of unique tweets coming from Twitter. One week of data reduces to a rather small sample data set, and while this may make the use of training and testing data infeasible for this project (but should be used whenever possible to better test the strength of a model), it does not prevent me from reaching a preliminary conclusion. Figure 4 below plots the data comparing unique tweets to the Nielsen rating.

Figure 4: The Relationship Between the Number of Unique Tweets and the Nielsen Rating



To first examine the question, I tried a simple linear regression through Python's scikit-learn library using unique tweets (in thousands) as my sole independent (feature) variable and the Nielsen rating (as a percent) as my dependent variable. This produces the following regression equation:

$$rating_i = 1.27 + 0.09 \cdot unique\_tweets_i ,$$

where i is a television program. The R^2 for this equation is roughly 0.05, indicating almost no relationship between the two variables. Additionally, this yields a root-mean-square error (RMSE) of 0.6 which may seem small, but is potentially large given the variables. While the coefficient was positive as I would have expected, there was essentially no relationship, contrary to what I expected. As a result, I realized I needed to add in some control variables to account for the various ways in which this relationship might be influenced. Primarily, I chose to control for both the network for which the show aired (since CBS is very older skewing and the CW is younger skewing) and the day of the week by adding dummy variables into the regression equation. Note that I excluded The CW and Wednesday from the regression equation as they are implicitly assumed in the equation. My resulting regression equation is as follows:

$$rating_i = 0.89 + 0.12 \cdot unique\_tweets_i + 0.90 \cdot ABC_i + 1.13 \cdot CBS_i + 0.31 \cdot FOX_i + 1.47 \cdot NBC_i - 0.96 \cdot Sun_i - 0.35 \cdot Mon_i - 0.44 \cdot Tue_i ,$$

where i is a television program. The R^2 for this equation is roughly 0.68, indicating a moderately strong relationship between the feature variables. Also, the RMSE decreases to 0.35, a nearly 50 percent decline indicating this new model is a better fit.

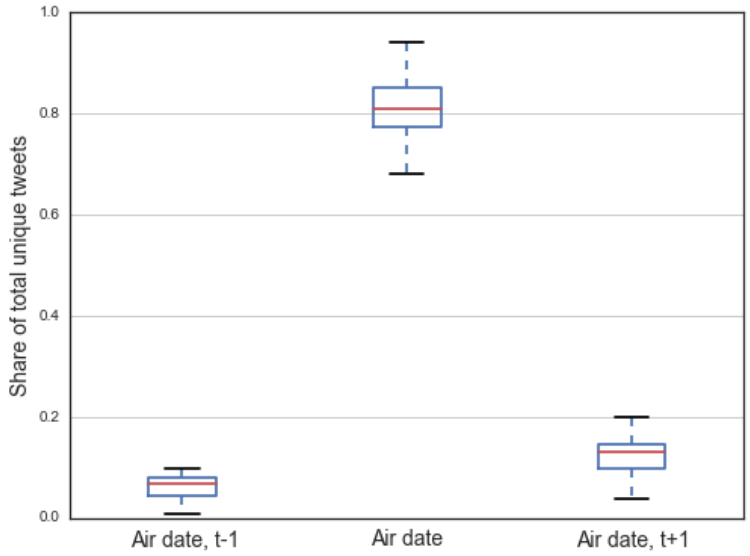
With more data and more observations, the accuracy of the model would improve (and allow for training and testing data to even further improve it). I would be able to add control variables for reality and scripted programming, and to differentiate reality programming between live sports events and other programs. As it stands, the model above yields a reasonable predictive model to evaluate the Nielsen rating based on the number of unique live tweets.

3.3. Other Visualizations

With the bulk of my analysis completed, I began attempting to model other relationships that might be worth looking into at some point in the future if I had access to the data. First, I wanted to see how Twitter activity for these programs differed on the air date versus the 3 previous and 3 following days around the air date. Given the data represents only a four day period, I chose to simplify this by using the days previous and post air date. Note that if a show airs on Wednesday, then I chose to call Sunday its "post air date." From Figure 5 below, it is shown that on average, 80 percent of a program's viewership occurs on the air date, with slightly more people tweeting on the day after versus the day before. This further identifies Twitter as a potentially good avenue to examine viewership because it indicates that when people are tweeting, it is on the same "live + same day" schedule that advertisers focus on for Nielsen. The main downside is that this does not show what the users are saying about the specific program. Some users could be using this as an avenue to claim how much they dislike the show and may

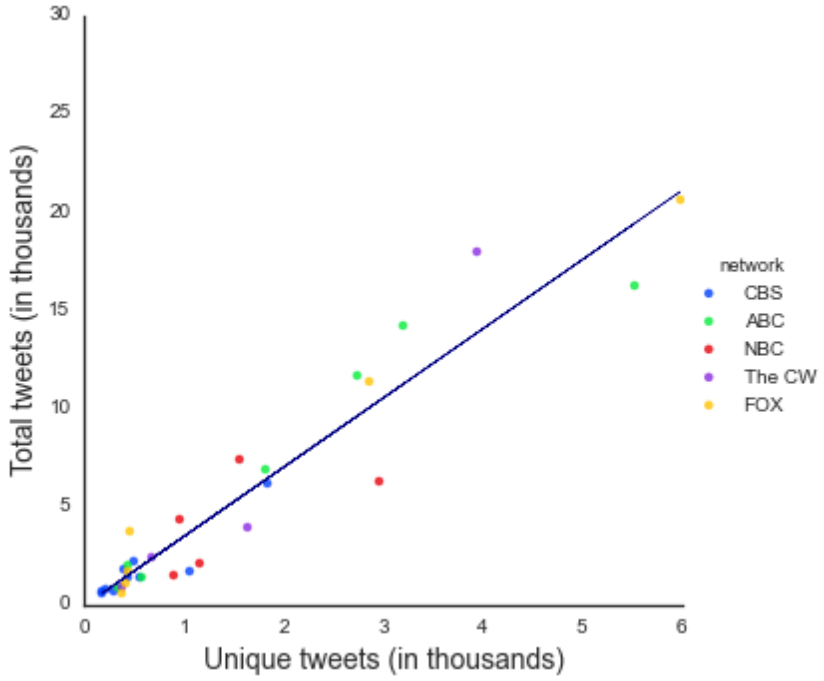
not actually be watching it. However, this is itself a double-edged sword as while it may encourage others to "falsely watch" a program (criticize it without actually watching) it may also remind others to turn on their TV and tune in. The best way to examine this would be some form of sentiment analysis on the tweets themselves and would be a great future avenue of exploration for this data.

Figure 5: Most Unique Tweets Happen on the Air Date



Lastly, it might be useful to consider looking at the "spamminess" of tweets, or in other words, the number of times a distinct user tweets during a specific "live + same day" period for a television program. The goal for this metric would be to try and predict the average age for each user so that advertisers would know how best to advertise given the average age group for a specific program. This would be one element to a larger model, though, and since demographic information is not available in Twitter, the dependent variable, which we would like to measure, would itself be a prediction. However, being able to develop this type of model would be a boon to advertisers and networks alike. Figure 6 below displays the number of unique tweets against the total tweets for a program. While the line in the figure is a simple regression line, it serves as an indicator so that those programs above the line are "spammy" and those below are not. Please note that the code used to produce the above visualizations can be found on my public GitHub under *twitter_analysis.py*. Additionally, each figure is located as a separate PNG file there as well.

Figure 6: Relevance of the "Spamminess" of Some Twitter Users?



Conclusion

I both expected and was surprised by the strength of the model since my usable data set was so small. The results make for a great start to what would be a mostly large-scale project given the resources required to analyze this data across a larger period of time. As a result, I learned just how valuable Apache Spark and operating on a cluster are to the strength of one's ability to analyze "big data." The amount of information coming from Twitter is massive, and given more time, I am sure I could think of new and interesting ways to capture information from it, especially since the text of the tweet went unused. One example might be to use the text of the tweet to examine user sentiment (positive and negative posts) and its relation to the ups and downs of the Nielsen rating. Developing a model to predict whether a Nielsen rating might rise or fall in the next week based on the tweet content opens the data up to a completely different set of models which could be used to gather more information. No one likes to see their favorite program go off the air prematurely, and being able to properly leverage other data sources such as Twitter to capture their viewing behavior might be the key to saving a show from cancellation in the future.