Towards Improved Sentiment Ratings

SOC 538 Term Paper

Ian Kennedy

12/11/2017

Introduction

I created a web application using RShiny to collect human-coded sentiment ratings of a sample of tweets about black Americans. I gathered a tweet sample, deployed the app on shinyapps.io and Amazon Web Services (AWS), and ended up with 361 ratings of 120 tweets. The full code of that app is included in the folder 'TweetRater' which was zipped along with this file. The ratings are saved in 'tweetratings.csv'. Details, including the full text, screen name, and id, of all the tweets used in ratings, are in 'fulltweettext.csv'.

What makes this interesting
Text analysis on large datasets poses many difficulties. One of these is sentiment analysis. Researchers hoping to see how views change over time or over the course of a document must have reliable and consistent methods for measuring those views. Sentiment analysis is a general term for using computational methods to discern subjective meaning from texts. Depending on the method, sentiment analysis algorithms can produce results that predict human ratings rather poorly. For instance, Flores (2017) shows that a simple lexical grader produced results that were little better than chance, while an improved method was in accord with a human graded sample only 64% of the time.

Long Term Goals
As a relative novice to the text analysis game, I opted to begin not by developing a sentiment analyzer, but rather to develop a usable test set for the purpose of comparing the strengths and weaknesses of existing methods. My long-term goals, then are to use my test set to find, tune, or build a sentiment analysis method that exceeds 64% accuracy on testing for anti-black racism. This goal intertwines with an intention to work on building an analysis tool that could be used more generally, say that could recognize the existence of racist sentiment and identify the target of that racism.

One purpose of this kind of project is to improve our ability to use large sets of text data, whether from social media, oral histories, or archives, to make sociological conclusions. This purpose is in line with Flores's work. Another goal is to, by analyzing this kind of discriminatory language at a large scale, understand more about the process of reproducing discursive discrimination itself.

Short Term Goals
In the shorter term. I aim to continue tweet-collection efforts until I have a larger sample of rated tweets, and then begin using the sample to test the sentiment analysis methods used by Flores, which he has graciously shared with me. Those methods rely on a lexicon of positive and negative words to produce a sentiment score, with the possible inclusion of basic topic modeling. In addition, I intend to attempt to apply basic machine learning techniques to compare those results with the traditional lexical methods.

Project Outline

This section covers the gathering of the tweet sample, the development and deployment of the app, and the process of collecting results.

Twitter API
I collected various samples of tweets over the course of two weeks to develop a sense for what sampling method would be best for the final ratings. Using Twitter's API and the 'searchTwitter' command from the twitteR R library, I searched for tweets which matched the search terms about black Americans listed by Flores in his appendix: 'African-American', 'blacks', 'AfricanAmerican', 'black people', 'black ppl', 'black guy', 'black girl', 'black dude', 'black men', 'black female', 'black man', 'black male', 'black women' (2017). I

added the terms 'race', 'racism', and 'white supremacy.' The sample was limited to these terms based on the hypothesis that computerized sentiment analysis would work better within a specified context. The final sample had 450 distinct tweets from 403 different users.

RShiny
The RShiny app was developed as a tool to display tweets in the sample and allow users to rate the sentiment of the tweet. Many aspects of development stretched my learning. I had never written a UI of any kind, let alone in RShiny, so I learned a lot in this process. The app initially stored the tweets, user information, and ratings locally as csv files. A main challenge was making sure that users continued rating tweets where they had left off in the sample. I used a dedicated file for each user to store the number of tweets they had rated, but this was a clumsy solution.

shinyapps.io
RShiny is paired with a free service called shinyapps.io, which will host small Shiny apps. However, server instances on shinyapps.io do not include persistent storage, so I couldn't store user information and ratings locally. Moreover, I wanted users to be able to create their own log in information. I used dropbox and google forms, along with R packages rdrop2 and googlesheets, to implement cloud storage of responses and user information. These worked well for my purposes, but they are slow and sometimes unreliable. In the future I would prefer to use a proper database solution. Shinyapps.io only included 20 hours of user time a month, and my alpha testing seemed to take up about half of that over the course of a few days. Since I didn't want to run out of server time while collecting ratings, I opted to set up the app on AWS.

AWS
Having never set up a remote server before, it took a few tries before I had a EC2 micro instance with Ubuntu, R, and Shiny Server installed and running. Had I gone this route initially, I wouldn't have had to work with dropbox or googlesheets and could've saved information locally. Given that all of that was set up, though, it was easier to just push my existing code to the server. Not surprisingly, there was lots more work to do to make sure everything ran smoothly in this new space, but eventually I got it working on AWS and Shiny. It was finally time to collect ratings.

Collecting Ratings
Because I was wary about the stability of my code, I tried to recruit only users who I contacted in person so that I could explain a bit about the project and make sure they knew that the app might crash or look weird. It turns out that was a real problem. While some of the crashing was due to inconsistency with AWS, often tweets failed to appear properly, and in some cases user input was not properly saved.
Users also complained about the difficulty of the task. Many users commented that it was really difficult to tell the intentions of the tweeter from the tweet-text alone. Others commented that many tweets didn't actually have to do with black people and those that did were mostly neutral.

Table 1: Distribution of User Participation

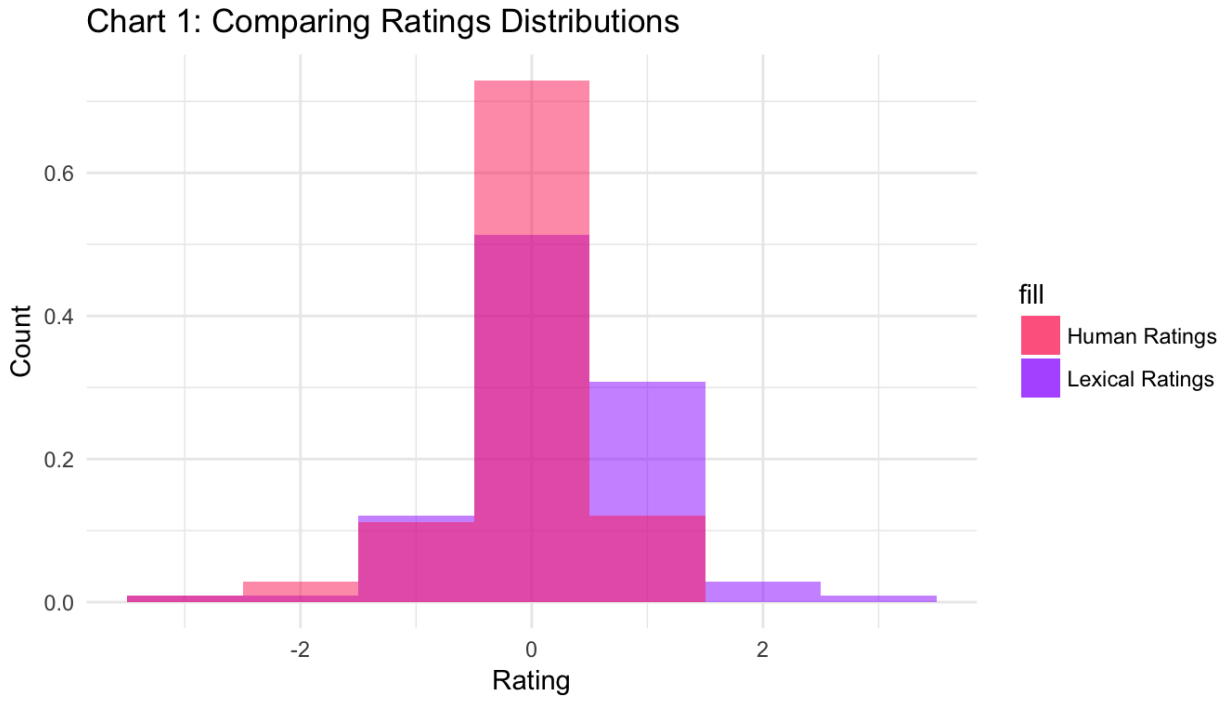
	Users
Rated less than 20 tweets	6
20 to 40 tweets	1
40 to 60 tweets	1
60 to 80 tweets	2
80 to 100 tweets	1

I hoped to gather around 6000 ratings using around 20 raters. It turns out that expecting raters to rate 300 tweets was quite unreasonably optimistic. I actually had 11 users who rated an average of 32.82 tweets each for a total of 361 ratings. While I had hoped to remove tweets from the set after they'd been rated 6 times, that code only ran every five hours, so some tweets had more ratings than that. 120 tweets were rated at least once, 33 were rated at least 4 times, and only 18 were rated 6 times. This result has been somewhat disappointing. However, now that I know a bit more about how the rater is working, I'm tempted to cast a wider net, leverage some crowd sourcing, and see if I can at least get a training set with more than 100 tweets.

Basic Analysis

Though I didn't collect as much data as I had planned, it still seems appropriate to offer a simple analysis of the success of my rater. This is partially in line with my goals for the project as a preliminary exploration of the strenghts and weaknesses of traditional lexical sentiment analysis.

First, It's worth emphasizing that my set of tweets was dominated by neutral sentiment. In this both the human coding and lexcial analysis agreed. Using the 120 tweets for which I have human ratings, I created histograms of the ratings for both the human ratings and the traditional lexical ratings. I rescaled the lexical ratings so they were on the same -3 to +3 scale as the human ones.



The take away from this chart is relatively simple: both methods rate a strong majority of tweets as zero, or neutral sentiment. Both methods are also unlikely to result in extreme scores. Compared to human ratings, however, the traditional sentiment analysis was more likely to give a non-zero rating. This makes sense given that all tweets with an odd number of words from the lexicon, and most with an even number, will get a non-zero rating.

To get a sense of howe well the two methods agreed with each other, I compared how often the two rating methods returned the same categorical response: negative, postivie, or zero. Overall, they agreed only 23.53% of the time. Of the three categories, they conformed the most on positive tweets (72.73), the least on negative tweets (32), with tweets not far behind (0). The correlaiton betwen the two ratings methods is relatively week β=0.15, and insignificant at α=0.05 (p=0).

To show this discordence, I graphed the human ratings, ordered from lowest sentiment to greatest, and then plotted the lexical ratings on the same graph. It shows us that the discord shown by the accuracy scores above is unorderd. This is largely because this lexical sentiment rating method is extremely unreliable for this application.

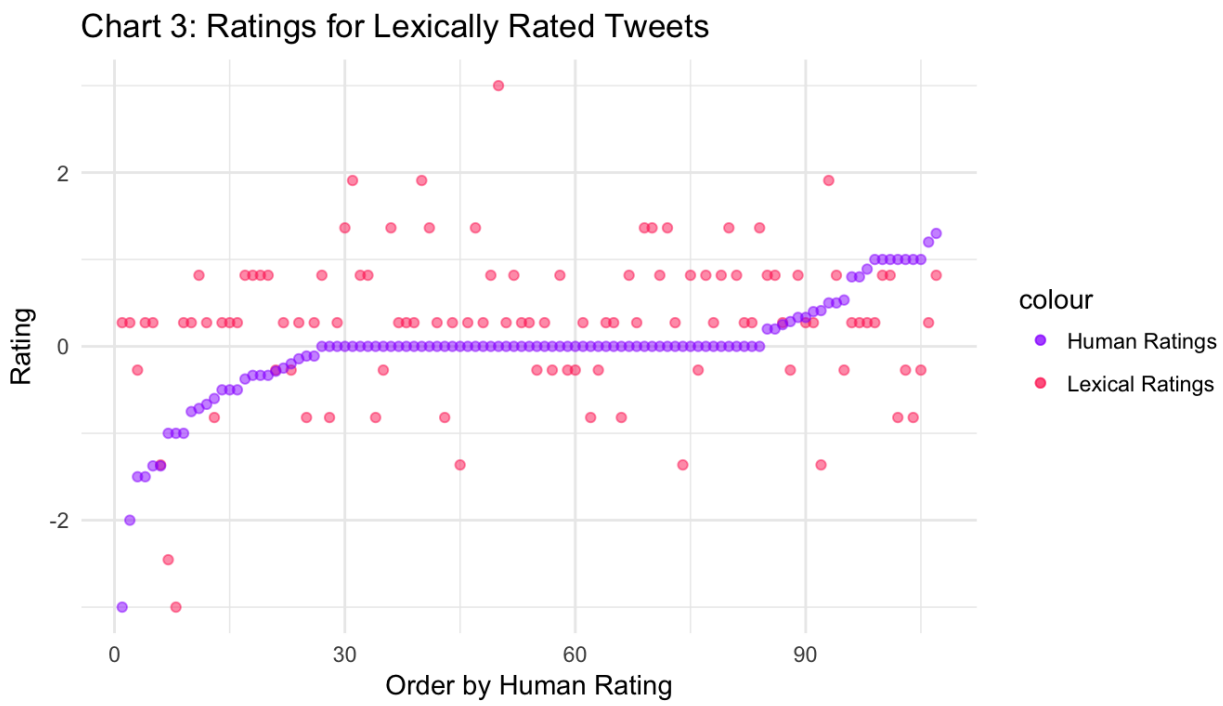
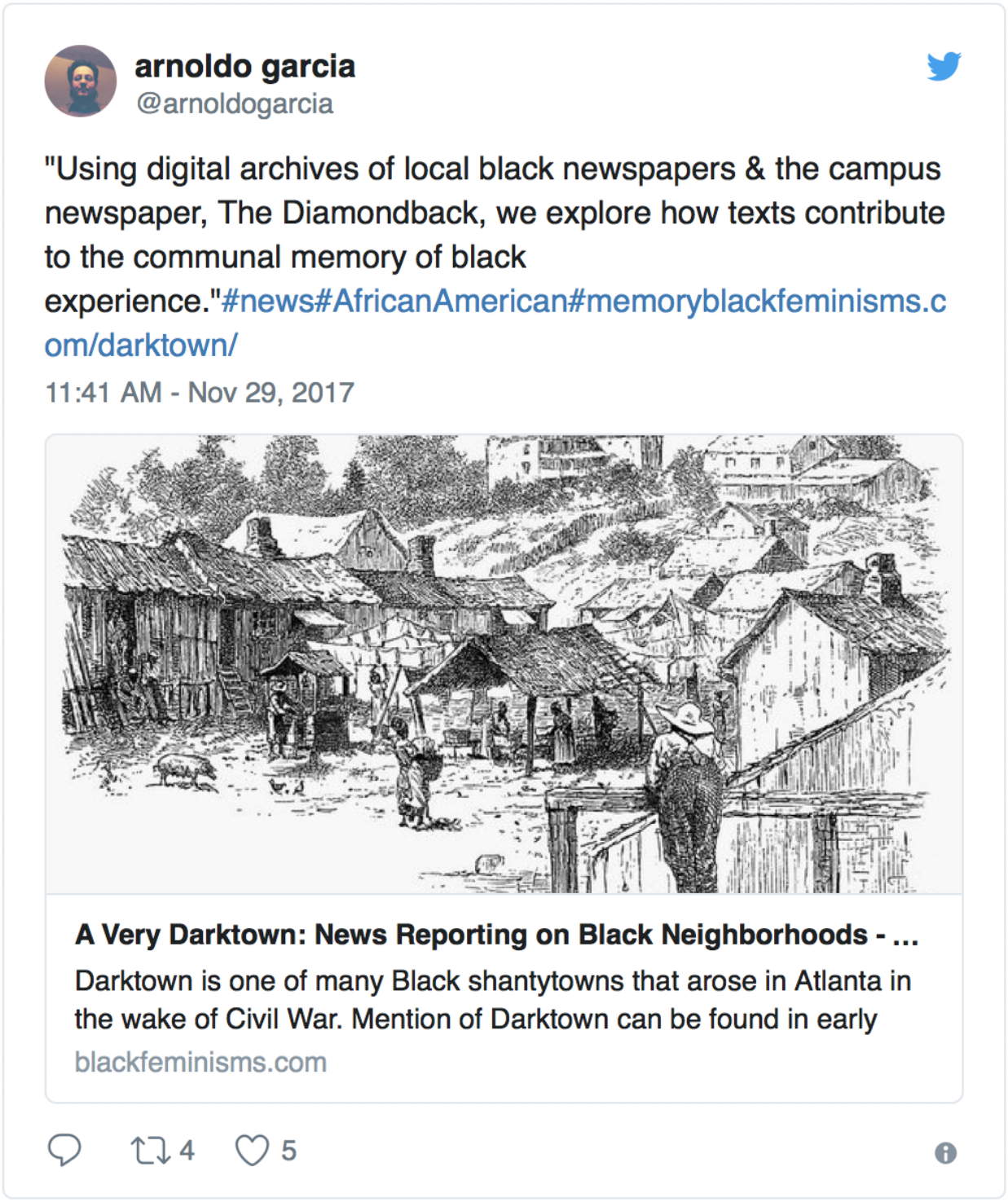


Chart 3: Ratings for Lexically Rated Tweets
Rating
2
0
-2
0
30
60
90
Order by Human Rating
colour
Human Ratings
Lexical Ratings

Brief Tweet Analysis

To understand a bit more about the difficulties of ratings, and becasue I can, I'll also look at a few cherrypicked examples of tweets.

Positive Tweet
Here's a tweet that was rated positively by the human raters:



{:width="250px"

My Rating: 1 Rater Rating: 1.2 Lexical Sentiment Rating: 0.27

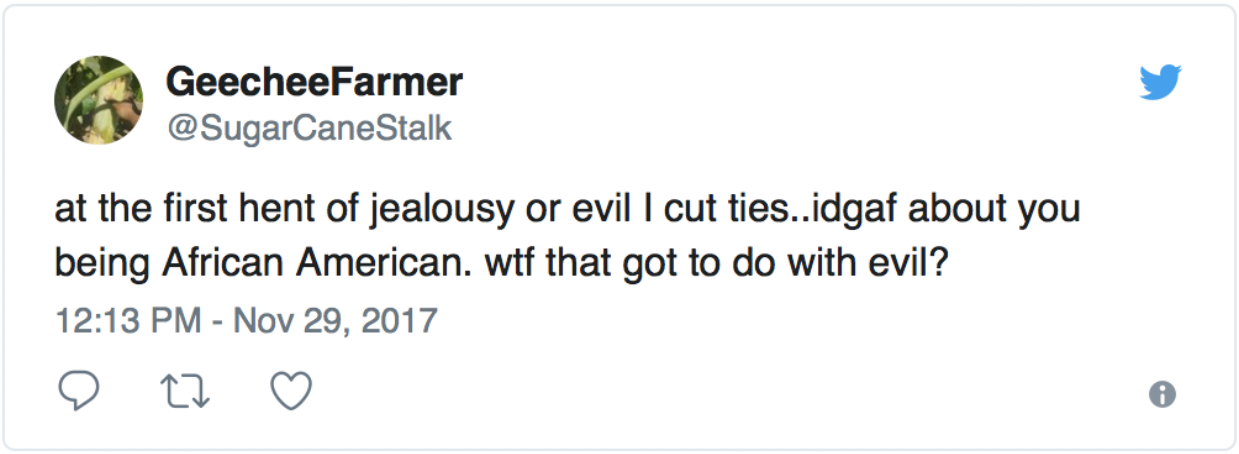
This is a great example of how difficult it is for lexical raters to pick up on contextual information. This tweet is positive about black Americans becasue it is part of efforts to produce more complety histories of the Unitied States. The human raters and I both rated this tweet positively for that reason, but the simple lexical rater left it neutral.

Negative Tweet
Here's a tweet that was rated negatively by the human raters:

{:width="250px"

My Rating: -3 Rater Rating: -1.38 Lexical Sentiment Rating: 0.27
I definitely thought this tweet was more negative than most raters, but it was still one of the tweets most consistently rated negatively. The negative sentiment comes from the tweeter's discomfort at having a black christmas toy. Agian, the lexical rater rated this tweet as neutral.

Large Difference Tweet
Here's a tweet that was rated negatively by the human raters:



{:width="250px"

My Rating: 0 Rater Rating: 0.41 Lexical Sentiment Rating: -1.36

This case goes the other way. Most tweeters and I read this as a neutral sentiment: the tweeter seems to think that most people would expect him to forgive jealousy or evil if his partner is black. He's saying that he won't do that. The lexcial rating picks up on the negative words in this tweet, but obviously cannot account for the context.

Conclusion

Ultimately, this tweet rating effort cannot be marked as a success. Not enough tweets were rated by enough people to form a sample large enough to use for machine learning. It also seems that human ratings themselves are not always reliable: there are a few tweets in the sample where I strongly believe that the human generated ratings are categorically wrong. All of those factors undermine the analyeses given above.

Future work to continue this project would probably first mean casting a much wider net for raters. I estimate that between 1/3 and 1/2 of all humans contacted to rate tweets actually rated at least one tweet. Since I contacted less than 50 people, increasing that group could dramatically increase the number of usuable ratings (the uncertianty here is driven by the fact that I'm not sure of the enrollment in 538/401). With even 100 human rated tweets with more than 6 ratings I would feel confident in the results of comparisons between sentiment raters. Alternatively, I could use what I've learned on this project to capture a new set of tweets and gather ratings using a tool like mechanical turk. While that would cost money, it might result in a more reliably useful sample.

That said, even with this small sample of rated tweets, I was still able to demonstrate the arbitrary nature of lexical sentiment scores in contexts like anti-black racism. I felt that SOC 538 acted as a kind of safe container for doing this kind of work which pushed me outside of my comfort zone as a coder and as a sociologist. I'm grateful for that. This project would have been impossible without lots of help from both Emilio Zagheni and Connor Gilroy. I am also deeply indebted to all of the raters, but especially James Kennedy, Angelina Eimannsberger, Hannah Lee, and Selen Guler who each rated more than 40 tweets.

Work Citied

Flores, René D. 2017. "Do Anti-Immigrant Laws Shape Public Sentiment ? A Study of Arizona ' S SB 1070 Using Twitter Data 1." 123(2):333–84.