Anime Ranking/Sentiment Comparison: Understanding Ranking and Review Relationship on Anime

Norlan Prudente
University of the Pacific, n_prudente@u.pacific.edu

1. INTRODUCTION

The goal of my project was to evaluate whether the reviews on some anime series, based on their sentiment, correspond with the actual rating that the user gave. The reason I got interested in doing this kind of research is due to my interest in anime. To be clear, I have found anime that have a high rating, yet their reviews do not seem to correspond with it. As a result, I used data science techniques to help me answer my question. I used myanimelist.net to gather my data for both the ratings and reviews.

My data came from a single source, which is the website called myanimelist.net. At first, I only wanted to get the rating and use Facebook and Twitter posts related to anime for their sentiment values. Unfortunately, Twitter only provided data for today's active anime and Facebook restricted me from accessing to those that are not in my account. Therefore, I ended up getting the reviews from one site and also used it to get the sentiments.

There are a few challenges I faced throughout this project. One of the challenges that I faced is how big the data is. The clean data consisted of 12gb of string data, which was 6 times for the raw data. This was a problem, because I could not load the whole data in Tableu. In addition, I was also unable to load it into a dictionary using json loader or panda. Another challenge that was presented was the velocity of the incoming data. The website changed its page content every time someone posted a new review. This was a problem, because duplication could occur during data gathering.

2. SOLUTION AND SCALABILITY

With the problem mentioned above, the solution I used is to do everything in one go. To clarify that, instead of gathering raw data and doing all the cleaning and analysing later, I did all of them while gathering the data. First, I used beautiful soup to scrape the data and clean it at the same time (see Figure 1). Next, I used the nltk package from Python to separate the review (see Figure 2). Thirdly, I used the same file, AFINN-111.txt that was used in twitter sentiment, to get a sentiment score for all of the words. Fourthly, I added them up and divided it by the number of words, mean value, and then added 5 to the solution. I added 5, because the rating scale is from 0 to 10, but the sentiment score used -5 to 5. Lastly, after getting all of the sentiment scores, which I stored in a dictionary, I wrote them into a csv file (see Figure 3). With this process, I was able to solve my big volume problem, because I did not have to reread the data and try to store them in a file. The process went from trying to save 12gb of clean data and performing analysis on it to just 225kb of sentiment scores. To solve the other problem about velocity, I visited the page backwards instead. Essentially, I went to page 1500 first and descended to the first page. This solution saved me from having duplicates. Of course, I missed a few reviews, but it is better than having duplicates that would throw my calculation off.

For the scalability of my program, it would be linear. The more cpu and internet speed I have, the faster my program proceeded. I am claiming this, because the internet can make my program run faster. That is to say, it is a web scraper and beautiful soup access each page via internet. As for the cpu, it will be faster, because it will increase the amount of resource or power I can use. Another option that I could use is parallel computing, but I do not have any experience on it.

```
#starting url
url = "https://myanimelist.net/reviews.php?t=anime&p=" + str(i)
#reset error
error = 0

#print the current page
print ("Gathering page " + str(i) + "\n")

#used for beautifulsoup
r = requests.get(url)

#get the whole html to be used by beautiful soup
soup = BeautifulSoup(r.content, "html.parser")

#html part that will be explored. This is what we're interested in
data = soup.find_all("div", {"class": "borderDark"})

#loop through all the data that was gathered
for item in data:
        try:
            #anime title
            animeTitle = item.contents[1].select("div > strong > a")
            #store it to our map as key
            if not animeTitle in animeReviews:
                animeReviews[animeTitle] = {}
                #will hold up the scores done with NLP
                animeReviews[animeTitle]['score'] = 0
                #count the reviews stored here to be used later
                #to find the mean
                animeReviews[animeTitle]['count'] = 0
                #will be used to store the mean
                animeReviews[animeTitle]['mean'] = 0
        except:
            continue
```

Figure 1. Sample code of how beautiful soup gathered and clean data.

```
#tokenize the review
tokenizeReview = nltk.word_tokenize(review)
```

Figure 2. Simple way to tokenize strings using nltk.

	A	B
1	Name	Rating
2		2.81922
3	Minami-k	1.4
4	Uchuu Kyc	2.366014
5	Ushiro no	0.051923
6	Forest Fai	1.037037
7	Yowamusl	2.090774
8	Bleach	1.577143
9	Ooyasan v	2.654855
10	Higurashi	0.773127
11	Tiger Masl	2.358929
12	Dark Blue	-1.76232
13	Zankyou n	1.062585
14	Shisha no	1.161548
15	Densetsu	1.188665
16	Tsuruhime	3.666667
17	Nabari no	0.980428
18	Dragonaut	-3.33333
19	One Piece	-1
20	Guy: Youn	-1.22581
21	Fune wo A	2.076605

Figure 3. An example of what the csv file looked like.

3. IMPLEMENTATION DETAIL

As for the implementation, I have tried and used a few tools. First is Python. Python is one of the few language I was familiar with. It has a wide choice of library that can be used for data science. We also used it for lab 1, Twitter sentiment analysis; therefore, it is what I used. I used nltk, csv, and beautiful soup to make my program easier. As mentioned above, beautiful soup was used to scrape data off myanimelist.net. As for nltk, it was used to tokenize the reviews that I got from the site mentioned above. Lastly, the csv was used to write the result to a file. I used csv, so that I could view them in excel. Secondly, I tried to use Tableu, but it did not provide what I wanted, which was to make a radial graph. Thirdly, I tried to learn d3,js, but sadly, I was not familiar with javascript, and the sample in the website that I liked did not provide sample code. Lastly, I used excel to make my graph. Excel had the perfect one I was looking for, which was to have two sets of data overlaying one another and showing the difference in them. A radial graph was the most suitable one that I found. It is able to clearly show that sentiment values are lower than the rating scores (see Figure 4). Overall, I ended up using python with the packages mentioned above to process my data and excel to visualize it.

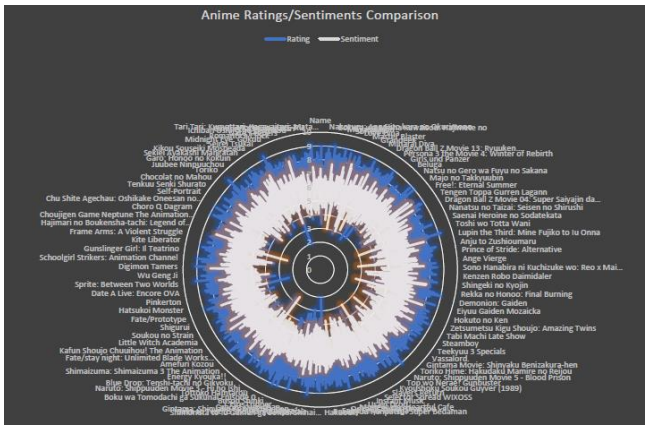

Figure 4. Sentiment value vs Rating

4. RESULT

The result I got is that the rating that users give are higher than the actual sentiment values of the rating, as mentioned in the implementation section and shown in Figure 4. My figure shows only a few of the most popular and active anime shows, but it analyzes more than 5 thousand anime. In my opinion, I have successfully answered my hypothesis and found a correlation in my data.

5. CONCLUSION

I have always wondered if the ratings and the reviews of the anime really correlated with one another. In conclusion, I was able to use data science to answer my curiosity. I used python and its library, beautiful soup, nltk and csv, to help me gather, clean, analyze, and store my result. As for the visualization, I used excel, because it is a simple, yet powerful, tool that was able to display my data accurately. As a result, I found that the sentiment values of the reviews were lower than the rating that the user has given them. Therefore, if you were to go and look for a good anime, you should read the reviews and judge on your own rather than basing it on its rating.