Yelp Recommendation Based on Aggregated Ratings from Sentiment and Stars

Justyn Lee
University of Southern California
Los Angeles, California
justynle@usc.edu

David Goodfellow
University of Southern California
Los Angeles, California
dgoodfel@usc.edu

Tu Mai Anh Do
University of Southern California
Los Angeles, California
tudo@usc.edu

ABSTRACT

Traditional recommendation systems used by Yelp are based off the 5 star rating system inputted by the users. However, this can reduce suboptimal recommendations due to the fact that users can be fairly biased where they disproportionally input an extreme value that is not reflected in the sentiment of the text content of the respective review. Our work attempts to combine the sentiment scores from the text content as well as their star ratings in order to create a new normalized rating value. This new rating will then be used in a collaborative filtering recommendation system which will be proven to recommend businesses that the previous system on solely stars would not.

CCS CONCEPTS

• **Data Mining → Recommendation Systems**; • **Sentiment Analysis**; • **Clustering**;

ACM Reference Format:
Justyn Lee, David Goodfellow, and Tu Mai Anh Do. 2018. Yelp Recommendation Based on Aggregated Ratings from Sentiment and Stars. In *Proceedings of University of Southern California (USC'18).* ACM, New York, NY, USA, Article 4, 5 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Over time, Yelp reviews have become more polarizing, with the percentage of one-star and five-star reviews blooming in comparison to the rest over the last 10+ years. [1] Conceptually, this makes sense as I am more driven to write a review if something was awful or amazing. However, many of these reviews may have bias. Therefore, it can be faulty to only show the star rating average for businesses and to use this for a recommendation system. In our work, we aim to create a new star metric that takes into account both the star ratings and the sentiment of the review's text content to create a more normalized and unbiased rating for businesses which can also be used in a recommendation system to create a wider domain of recommendations to present a respective user with.

Our data comes from the Yelp Dataset from Kaggle.[2]

USC'18, December 2018, Los Angeles, California USA

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

2 METHODOLOGY

Our approach to completing this system involved two parts. The first phase was a proof of concept phase involving a smaller subset of 10,000 reviews. This first phase was paramount for our project as it allowed us to validate our new process as valuable and statistically different from traditional systems. Using this set size also allowed us to fairly quickly determine this as we did not have to develop highly efficient scripts through Apache Spark.

Once the proof of concept seemed promising, we ended up running a larger comparison on approximately 600,000 reviews with a total memory size of 500 MB. We originally wanted to run on the full review set (4.7 GB) but this ended up not being feasible due to lack of RAM space on our local computers and due to an issue with running that large of a set on our sentiment analysis system which will be discussed later.

We will describe each of our phases and the results we got from them in more detail below.

3 PHASE 1: 10,000 REVIEWS

Each of the two phases consist of similar steps. First, sentiment analysis was performed on the text reviews. Second, statistical analysis was done to compare, visualize, and describe the sentiment and the normal stars. Third, a new aggregated score was iterated on and finally formed. Fourth, clustering and other visualizations were performed on the traditional star scores and our new aggregated score for insights and validation. Lastly, the new scores were fed into a recommendation system whose results were compared to the traditional recommendations based on stars.

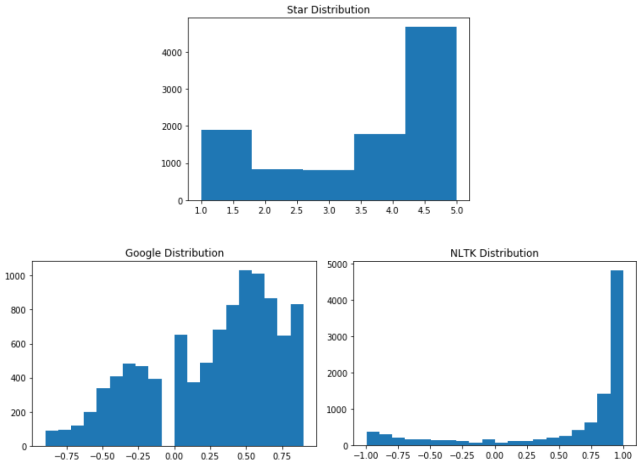
3.1 Sentiment Analysis

There are a few well-known APIs which can generate sentiment scores for a sentence or paragraph. The two we employed in the Python scripts in this step are the Natural Language Toolkit (NLTK) and Google Natural Language API. NLTK returned four features: negative (0-1 range), neutral (0-1 range), positive (0-1 range), and compound (-1 to 1 range). The Google Natural Language Sentiment returned two features: score (-1 to 1 range) which is the equivalent of compound in NLTK and sentiment magnitude (positive integer) which describes the quantity of words/phrases that enforce the score given.

We decided to move forward and compare the NLTK compound score and the Google sentiment score for a few reasons. For NLTK, the negative and positive values were typically both relatively small with the neutral being near the higher end of the range. We thought finding a function to aggregate these would not be better than the compound that is already given. For Google, we did not move forward with the magnitude because it would not only be hard to

create a function around, but moreover, it would be biased towards longer reviews as compared to shorter ones. Lastly, each of these two scores had the same range, making it easy to compare.

From there we compared the distributions of scores given by each in a histogram. These distributions revealed that a disproportionate amount of the NLTK compound scores were near the two ends of the range as compared to the Google sentiment score which was more evenly distributed. As stated previously, Yelp star reviews are already disproportionately distributed at 1 and 5 stars.

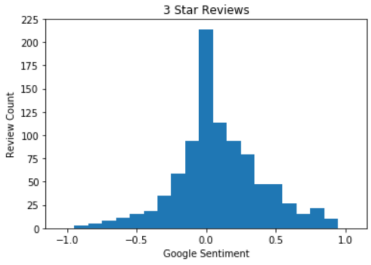


Due to these distributions, we began to lean towards Google Language API. However, before making the final decision, we wanted to conduct a sanity check to ensure the scores we were getting we relatively agreed with. We noticed that Google's results seemed to be more in line to what we would score a review's sentiment as. In fact, the NLTK had a few reviews that it had as slightly positive despite being fairly negative to us.

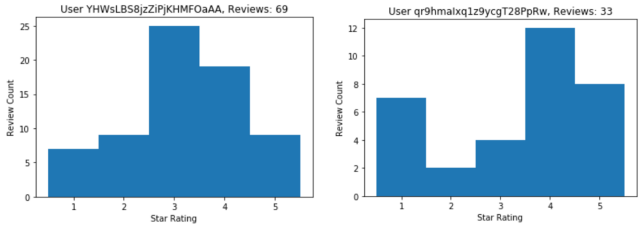
Due to these two reasons, we decided to move forward with the Google sentiment score as this would allow us to more intuitively create a new normalized aggregate score.

3.2 Statistical Analysis

Upon analyzing stars from the 10,000 review sample, a couple things became apparent about the data set which validated our initial concerns about biased 5 star reviews. First, as mentioned in the introduction, the distribution of stars from user reviews is heavy on the tails. Second, the sentiment scores vary wildly across reviews that are counted equally under the traditional 5 star rating system as seen below.

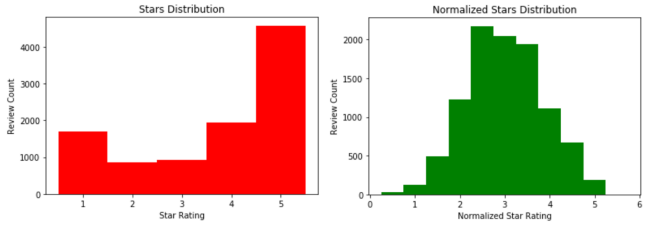


These two observations reveal that the meaning behind the traditional Yelp rating system is flawed and does not accurately reflect the true "goodness" of a business based on user feedback. The average rating of a business is not based on a normally distributed pool of star ratings, rather the rating is the average of mostly excellent (5-star) and terrible (1-star) reviews with little from the middle ratings. But even the star ratings themselves should not be taken at face value because behind each star is a text review which typically reveals a more nuanced, if not contrary, rating beyond just five integers.



The last observation upon statistical analysis is that there are different kinds of users in terms of how they rated businesses. On the one hand, we observe a more-or-less normally distributed set of star ratings (above left); whereas on the other hand there is a bimodal distribution that has a higher normal distribution mixed with a lower normal distribution (above right). In these two cases, a 3-star rating means different things to each of these users and should not be considered to be the exact same rating.

3.3 New Scores



Given the knowledge gleaned regarding the data set from the statistical analysis, we constructed a new, normalized star rating. The algorithm involved two steps:

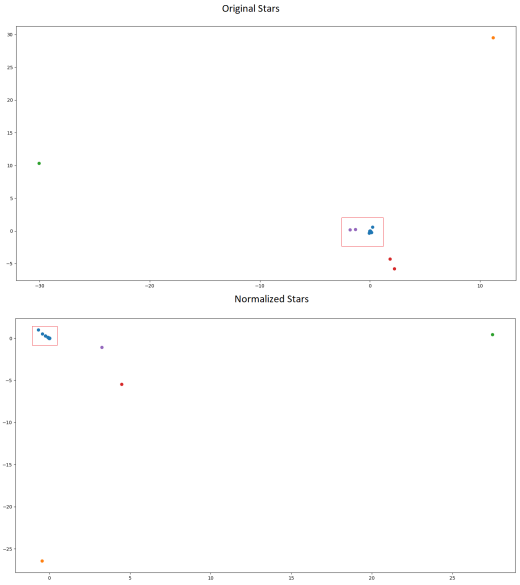
(1) Normalize by sentiment. We created a "sentiment star" by nuancing the original star based on the sentiment of the text review. This helps account for the differences in equal-starred reviews which have varying degrees of positivity reflected in the words written for the review.
(2) Normalize by user. We used the"sentiment star" and each particular user's distribution (see Section 3.2) to determine a normalized star relative to the reviewer. This reduces bias from "easy" vs. "harsh" reviewers and puts them on a more level playing field.

The goal of these normalized star ratings is to properly reflect the relative meaning behind a review based on each user, so that a normalized 3-star actually means "average", a normalized 5-star indicates an "excellent" business, etc.

3.4 Clustering

We want to cluster the users that are similar based on adjusted rating so that we might be able to group similar users in the same group and leave biased users on other different groups. With such an expectation, we make the comparison between clustering results using normalized stars to the result using original stars in the Yelp Review dataset.

First, we ran the K-Means clustering on users for the first 10,000 reviews to obtain an overview insight of clustering with a smaller dataset. Since the clustering is made on users, all the reviews are first grouped by the user who made each review and then a sparse vector of all occurring businesses is generated. Each vector represent one user's ratings across all businesses. Hence, the clustering using these vectors as input data will reflect the correlation of users on rating.



The results always have one big cluster that contains most users, which is probably because many users only rate on a small number of businesses. In order to better visualize the clustering of the giant group, we decided to use Principal Component Analysis (PCA). The goal is to reduce the number of dimensions of the sparse vector of user ratings to smaller number of dimensions, especially here is 2 dimensions to visualize the users on a 2D scatter plot. We computed the principal components first and then ran the K-Means on these components. In the figure above, the big cluster is covered in a red rectangle. Even though the star has been already subtracted by the mean rating to avoid the misleading information where businesses with no rating could be incorrectly considered as a rating of zero, the results did not become better.

3.5 Recommendation System

Our recommendation system is a user-to-user collaborative filtering model built in Scala. The baseline comparison metric for a recommendation system is root mean square error (RMSE). We also wanted to produce a way to see the similarity, or intersection, of each set. This is important because the RMSE comparisons for each do not directly tell the similarity between each as the ground truth for the first system is the star ratings and for the new system it is our new aggregated rating. In order to do this, we find the intersection of those user-review combinations that have an RMSE of under 1 and finding the intersection percentage of these in the test set. Having an RMSE of under 1 for a specific review means the recommendation was quite accurate in its ability to predict the user's score. If the intersections of each are extremely similar, this would not validate our new model as it does not create a new domain of user-business recommendation tuples.

For our first 10,000 items we broke up the train and test set into 9,000 and 1,000 reviews, respectively. The RMSE of each was 3.62 for the original system and 2.92 for our new system. These are both extremely high considering the upper range for this is 5. However, this can be attributed to the small size of the test set and the sparseness of it as described previously. In finding the intersection of those under 1, only 35% of the error under 1 in the star set were also in the adjusted set. This made us believe there was validity in our new system but this could not be proven on such a small set.

4 PHASE 2: 600,000 REVIEWS (500MB)

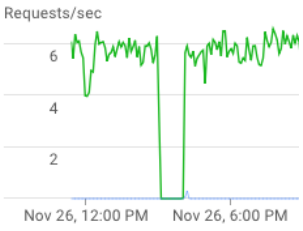
The results from Phase 1 were promising and gave us validity in our general approach to the problem. For Phase 2, we open up our five-step approach to a 500 MB set of reviews. In order to process this sized batch, many of the steps were re-engineered to utilize the advantages of Apache Spark.

4.1 Sentiment Analysis

As previously mentioned, we desired to run all the reviews in the yelp review set (4.7 GB) through. This means we would have to run each of these through our Google Natural Language API. This ended up being an issue for a few reasons.

The first issue deals with the pricing of hitting this API. The price is $1 per 1,000 hits. Google Cloud provides students with a $300 free credit to use. Between the three of us, we had a sum total of $600 credits due to a few of us using their APIs previously. This allows us only to retrieve 600,000 reviews for free (500 MB). If we were to scale this to the full set, it would cost approximately $5,040 out of pocket (4.7GB - 0.5 GB = 4.2 GB. 4.2 GB / 0.5 = 8.4. 8.4 * $600 = $5,040.)

The second issue deals with the time necessary to process this many requests. As shown in the image below, on average when up and running, 6 sentiments are completed per second through the Google Cloud Natural Language API. At that rate it would take about 27 hours and 45 minutes to process 600,000 records (600,000 records/6 per second/60 seconds in a minute/60 minutes in an hour = 27.77 hours). If we were to scale this to the full dataset it would take over 10 days which was not a feasible timeline.

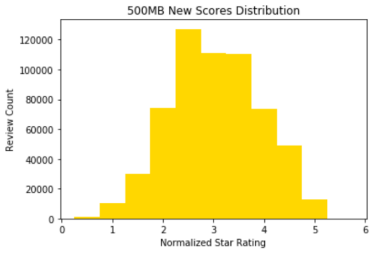


From this we decided to move forward with 600,000 records. For running it through the API, we programmatically broke up the set into sixty 10,000 review chunks and ran them one-by-one through the API and uploaded the results for each in our shared folder. Doing it this way ensures we make progress if any errors are faced and also gave us an audit trail to see how far in the process we were at any point.

4.2 Statistical Analysis of New Scores

Due to the larger dataset size of 500MB, we also ran into some memory limitation issues when adapting the original stars to the new star scores. The algorithm depends upon constructing sample t-scores for each user's reviews, which did not have a built-in Spark function for slices of the data by user. Rather this calculation required storing intermediate data in maps for each user, which overwhelmed the spark process running on our local computers on the larger dataset.

Instead of calculating t-scores for each user on the entire 500GB data set, instead we calculated new scores for each user for each 10,000-review slice that was mentioned in the previous section. These new scores were then aggregated into a full distribution for all 600k reviews as seen in the figure below.



Both the visual inspection and the summary statistics for this distribution reflect that it is approximately normal with a mean of 3.0334 and a standard deviation of 0.9016. This places 65.7% of the scores within one standard deviation and 98.0% of the reviews within two standard deviations. These percentages roughly reflect the 68-95-99 distribution percentages characteristic of normal distributions and validate the normalcy of the new scores centered around 3. Therefore, by using these new scores a business's reviews will accurately reflect how good the business is compared to the average business.

4.3 Clustering

We were unable to run PCA on this larger set because of computing limitations. Calculating principal components over rating vectors is an computation-intensive task whose computing time depends on the the number of principal components and the number of vectors. As the dataset gets larger, the number of occurring businesses also increases. For example, at 600,000 reviews, the number of businesses is 122,011 and the number of users is 245,271, which makes PCA computing turn out to be a bottleneck as this computation has to reduce the number of dimensions from 122,011 to 2. As a result, this bottleneck prevents us from applying the clustering with PCA at this size.

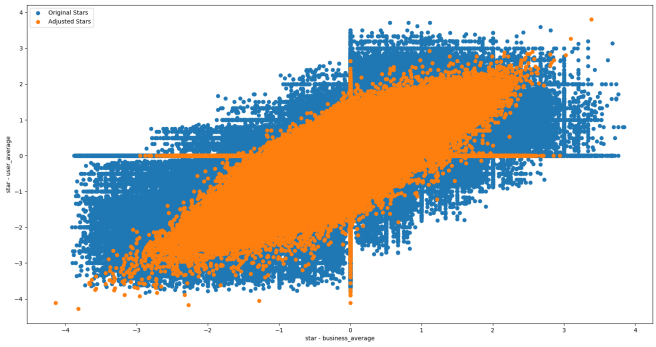
Similar to the issues we faced when clustering on the 10,000-review set, when we ran the K-Means on the pure rating vectors without using PCA with the dataset of 600,000 reviews, the result of clustering again is a giant cluster. This goes against our belief that was at large amount of data, since the number of users and business increased, the result of clustering is improved.

This observation opens a new requirement and the necessity for a new approach in order to address the adjusted ratings' evaluation. The information regarding such a new approach for evaluating the new scores is described in Section 5.

4.4 Recommendation System

The same process was completed as before for the recommendation system. For the 600,000 set, we used an 80/20 split. The RMSE of each was 2.77 for the original system (with 37,138 reviews with error under 1) and 1.64 for our new system (with 46,313 reviews with error under 1). Now, approximately 60% of the reviews with error less than one also were for adjusted. This amount of overlap is expected but it still creates a much larger domain for additional recommendations by widening it from 37,138 to 55,195.

5 RESULT EVALUATION



The aforementioned challenges form the necessity of a new approach to evaluate the new scores. For the larger set, we moved to a new scatter plot that compared the scores relative to both the user and the business. In the plot above, along the x-axis is the difference between the score and the business's average, while along the y-axis is the difference between the score and the user's average. If we look specifically at quadrants II and IV, the original stars cover a wider area than the new scores. Points in these quadrants indicate reviews in which the rating relative to the business's average is opposite to its relation with the user's average. For example, if a rating is less than the business average by 1 star, but is greater than the user's average by 2.5 stars that is an indication of possible positive bias in the original stars. In this way, the scatter plot provides evidence that the new adjusted stars (orange) are less biased overall than the original stars (blue).

6 CONCLUSIONS

Yelp's traditional 5-star rating system notionally seems like a good system to differentiate businesses based upon user feedback. However because of the rigidity of using only five integers which do not always accurately reflect the true sentiments of the user's review, we devised a new rating system which reduces user bias, normalizes ratings to a proper scale, and incorporates text mining analysis

from what the user actually wrote. This new rating scale is also more readable because a rating of 5 actually means the business is excellent, whereas a rating of 5 is severely diluted in the original stars by the fact that 5 stars is by far the most common rating left by users.

We validated our new ratings by creating a collaborative filter that predicts user ratings with 41% better accuracy than a collaborative filter using the original stars, and provides an additional 48.6% of accurate recommendations. Therefore, by using our new ratings, Yelp could significantly increase its ability to suggest new restaurants to users, which would result in higher advertisement effectivity and greater profits.

The next step would be to understand the characteristics of the additional recommendations that the new ratings produced. Based on our analysis, we would hypothesize that these particular users have some sort of bias that is corrected for by the new ratings. This could lead to more insights into sources of bias and how we can further adapt our ratings to reduce even more bias from the original star ratings.

7 APPENDIX

GitHub URL: https://github.com/tumaianhdo/inf553-adjusted-ratings
File structure:

/ dataset-extraction (Goodfellow)
/ sentiment-analysis (Goodfellow)
/ normalized-stars (Lee)
/ stat-analysis (Lee)
/ clustering (Do)
/ visualization (Do)
/ recommendation-system (Goodfellow)

REFERENCES

[1] M. Woolf, "The Statistical Difference Between 1-Star and 5-Star Reviews on Yelp", minimaxir | Max Woolf's Blog, 2018. [Online]. Available: https://minimaxir.com/2014/09/one-star-five-stars/. [Accessed: 30- Nov- 2018].
[2] "Yelp Dataset", Kaggle.com, 2018. [Online]. Available: https://www.kaggle.com/yelp-dataset/yelp-dataset.