CS224w Final Report: Community-Based Yelp Personalization

Hao Zhang, Haoran Li and Pei-Chun Chen (Group 32)

I. Introduction

People use Yelp to search for everything and get recommendations. When we are reading a review, we want to know if the review itself and the reviewer are credible. Thus, it helps if we can know the credibility of the reviewer. In turn, a reviewer might be motivated to write more high-quality reviews. The user scores can be incorporated into calculating new (and hopefully more accurate) scores for the businesses. In addition, people in different communities have very different opinions about a single business. For example, a community of Chinese elderly people and another one of American youngsters can have totally different comments for a Chinese restaurant. However, Yelp currently only offers one single score for a business. With the goal of better reflecting different tastes from different communities, we want to generate communities from all users and put users that have similar tastes into one community. Then for all users in one community, we will put some bias on the new score that we generated for each business based on the other users review and their credibility in the same community. In this way, score of a business is personalized to each community. This community-based score is expected to be most informative and can be utilized to make personalized recommendation.

II. Prior Work

There are two main parts in this project. First, we calculate credibility scores for users. Second, we personalize the business stars shown to a user according to which community the she is in. In order to do so, we have to identify communities. For the first part, we found limited related work. There is research on implicit user credibility extraction for reputation rating mechanism in B2C e-commerce [4]. However, the conceptual framework of the mechanism is based on the source credibility model in consumer psychology, which is not very referential for our project. There is plenty of work about if a review helpfulness evaluation, for example, [7] analyzed how the perceived helpfulness of an Amazon review depends on its relation with other evaluations of the same product and showed the level of individual variation in star ratings for products matters. Nevertheless, we want to evaluate the user directly instead of the reviews she wrote. Thus, the bipartite structure between users and businesses is similar to the hub and authority set-up in HITS [3].

For the second part, we found different models for community detection. In [6], a set of algorithms for discovering community structure in networks, i.e. natural divisions of network nodes into densely connected subgroups, are studied. The algorithm involves iterative removal of edges from the network to split it into communities. In [5], a hierarchical agglomeration algorithm for detecting community structure is presented. This algorithm is faster compared to many other algorithms and applies the algorithm to the analysis of a co-purchasing network from Amazon.com to evaluate the algorithm. We use hierarchical agglomeration to do clustering on users and extend the algorithm described in [5] to weighted graph. We use some additional data structures to maintain changes in modularity to speed up the clustering process. We also explore the distribution of community sizes when partitioned at the point of maximum modularity.

III. Data

We will be using the Yelp Phoenix Academic Dataset provided by Yelp for the Yelp Dataset Challenge. The detail can be found here [1]. The dataset includes a generous sample of Yelp data from the greater Phoenix, AZ metropolitan area with $11,537$ businesses, $8,282$ check-in sets, $43,873$ users and $229,907$ reviews. Specifically, the dataset includes four json files:

- Business: the geographic info, category, review count and star rating for businesses.
- Review: the text, date, author, target, and votes (useful, fun, etc.) of reviews.
- User: the review count, average stars, and votes given by users.
- Check-in: the target business and the counts of check-ins at/on different times/days

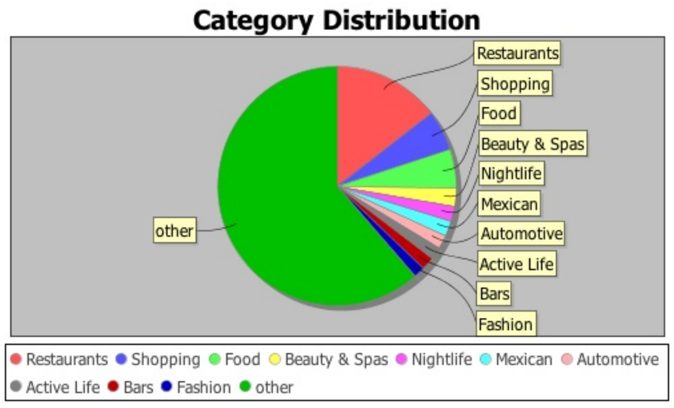
The authors are with the Department of Electronic Engineering at Stanford University, CA, 94305, United States. Email: {hzhang22, aimeeli, peichun2}@stanford.edu.

IV. Data Statistics

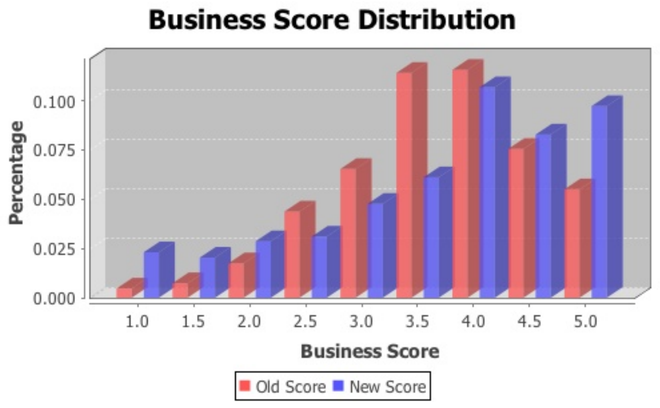
We explored several statistical properties of the data, which will give us an overview of how the data is distributed and help us come up with useful features.

A. Category Distribution

There are 508 categories of businesses in total, from the category distribution plot a we can see that restaurants constitute a large proportion of all the businesses and the rest are evenly distributed among all kinds of categories.



(a) Category Distribution



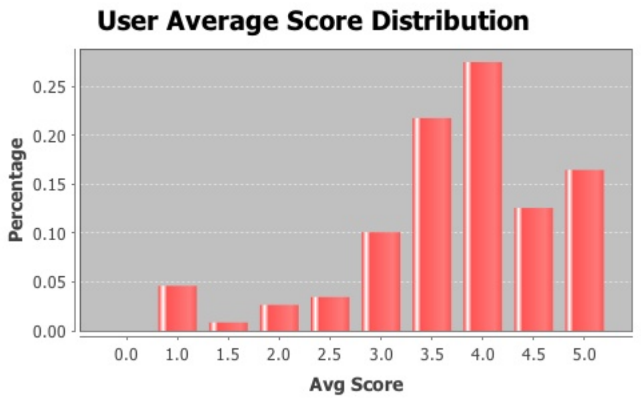
(b) Business Score Distribution

B. Business Score Distribution

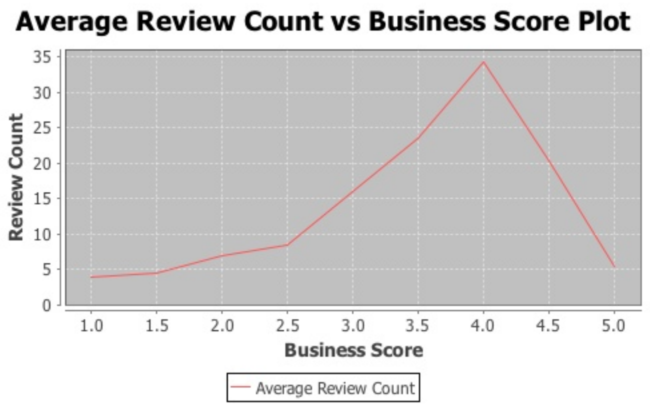
The average score of all businesses is around 3.67. The plot b includes distribution of original score and the new score that we give for all businesses. From the plot b we can see that the original score of businesses follow a Gaussian distribution. Most scores lay in the range of 3.5 to 4.5.

C. User average star distribution

The plot c shows the distribution of average score a user gives for all the businesses that he or she reviewed. Most users tend to give a score between 3.5 and 4. A few users have average score of 1 and 5, which means those users tend to give an especially low or especially high score and these users might have low credibility.



(c) User Average Score Distribution



(d) Average Review Count and Business Score Plot

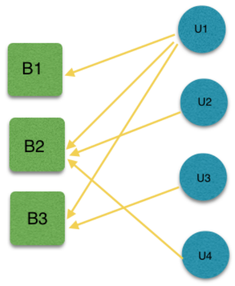
D. Review count and business score correlation

For each business score, we plot the average review count of all businesses with that score. From the plot d, we can see that higher scores tend to have more review counts, which is reasonable since more people will tend to go to businesses with high score. However, perfect score tend to have fewer review counts. One possible reason could be that not many people will give a business perfect score. Therefore, those with perfect score tend to have fewer reviewers.

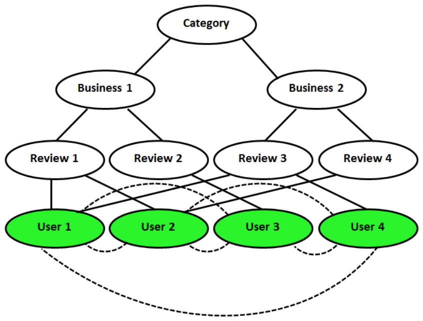
V. User Credibility Score

A. HITS Based Scoring Algorithm

In this method, we are using an algorithm similar to HITS to obtain the user credibility and business new star. Here, the user and business are acting as the hub and authority roles in the HITS algorithm. The structure of users and businesses is like e and the tree structure of the total system is like f. We firstly calculate the user



(e) User Business Relationship



(f) Structure of Yelp Network

credibility from the businesses every user has reviewed before. Suppose, the $\beta_u = \{b, where\ u\ has\ reviewed\}$ is the business set every user has reviewed. SU_b^i is the score user i gives to business b. And S_b is the score of business b calculated before or the original Yelp star for the first time. The user credibility of user i is

$$U^i = \frac{1}{1 + \sum_{b \in \beta_i} (SU_b^i - S_b)} \tag{1}$$

The $+1$ in the denominator is used to smooth the user credibility in case of the situation that all the $SU_b^i = S_b$ We can thus get a user credibility of the user.

Then we will calculate the business new star according to the updated user credibility. Let's suppose ζ_b is the set of user u who has reviewed business b. Then we can get the new score of the business by the weighted score of users who has visited the store

$$S_b = \frac{\sum_{u_i \in \zeta_b} U_i \times SU_b^i}{\sum_{u_i \in \zeta_b} U_i} \tag{2}$$

Before moving on, another step needs to be done is to normalize the user credibility to make it with a maximum range of 5. We can do it in such way:

$$U^i = \frac{U^i}{\max_{j \in \nu} U^j} \times 5 \tag{3}$$

Here the ν is the user set.

This process is operated in an EM manner, and the stopping criteria now is that for 95% businesses, their new star changes is less than 1%. And the results can be seen from figure b.

The algorithm can be seen at the algorithm 1 chart.

Algorithm 1 HITS Based Scoring Algorithm

```
Data: User Set and Business Set
Result: Find the user credibility and new business star
Build set of "Users" and "Business"
while True do
    Business to User
    for user in Users do
        TotalDifference = 0
        for review by this user do
            totalDifference += score difference to a business by this user and the star of the business
        end
        userCredit = 1.0/(1.0 + totalDifference)
    end
    User Credibility normalization
    User to Business
    for business in Businesses do
        TotalWeight = 0
        for review by this user to this business do
            newStar += userCredit * userScoreToBusiness
            TotalWeight += userCredit
        end
        newStar = newStar/(TotalWeight)
    end
    Scale User Credibility to a max of 5
    Stopping Criteria
    if 90% business stars have changed less than 5% then
        Break
    end
end
```

TABLE I: Feature Table

Feature	Weight (Old)	Weight (New)
Average stars	1.26	4.21
Review counts	0.83	0.41
Funny votes	0.21	3.06
Useful votes	5.12	3.59
Cool votes	-6.89	1.67
Total votes	-0.35	1.39
Score difference	-0.35	3.90
Absolute score difference	0.18	-40.28

B. Feature Based Scoring Algorithm

In this approach, we first identify useful features that help decide a users credibility and then use supervised machine learning models to find the feature weights. Specifically, we identified 8 features:

- Average stars: The average stars this user gives to businesses.
- Review counts: The number of reviews written by this user.
- Funny votes: The number of funny votes received by the reviews written by this user.
- Useful votes: The number of useful votes received by the reviews written by this user.
- Cool votes: The number of cool votes received by the reviews written by this user.
- Total votes: The total votes received by the reviews written by this user.
- Score difference: The sum of the star difference between the final business stars and the stars given to those businesses by this user.
- Absolute score difference: The absolute sum of the star difference between the final business stars and the stars given to those businesses by this user.

One difficulty here is that in order to use supervised models, we need the labels (true credibility values), which we do not have. Therefore, we introduce a second-layer into our model and use the stars of the businesses as the labels. Specifically, the original business score is calculated from the scores given to it by users as in

$$b_1 = \frac{\sum_{i=1}^{n} u_i}{n} \tag{4}$$

Our assumption is that after we apply the credibility score to each user, the new calculated business score should stay the same, as in

$$b_{1,new} = \frac{\sum_{i=1}^{n} c_i u_i}{n} = b_1 \tag{5}$$

The credibility score for each user is calculated from the features and their corresponding weights in

$$c_i = \sum_{j=1}^{m} w_j f_j \tag{6}$$

The features we come up with and the descriptions are listed in Table I. After inserting equation (6) into equation (5), we get the relation between the weights of features and the business scores in

$$\begin{aligned} b_1 &= \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} w_j f_j u_i}{n} \qquad (7)\\ &= \left[\begin{array}{ccccccc} f_1u_1 & \cdots & f_mu_1 & \cdots & f_1u_n & \cdots & f_mu_n \end{array}\right] \\ &\times \left[\begin{array}{ccccccc} w_1 & \cdots & w_m & \cdots & w_1 & \cdots & w_m \end{array}\right]^T \\ &= \left[\begin{array}{ccc} f_1(u_1+\cdots+u_n) & \cdots & f_m(u_1+\cdots+u_n) \end{array}\right] \\ &\times \left[\begin{array}{ccc} w_1 & \cdots & w_m \end{array}\right]^T \\ &= \left[\begin{array}{ccc} f_1U_1 & \cdots & f_mU_m \end{array}\right] \times \left[\begin{array}{ccc} w_1 & \cdots & w_m \end{array}\right]^T \end{aligned}$$

Expanding from one business to L businesses, we get the matrix representation $Aw = B$ in equation below:

$$\begin{bmatrix} f_1u_1 & \cdots & f_mu_1 \\ f_1u_2 & \cdots & f_mu_2 \\ \vdots & \ddots & \vdots \\ f_1u_L & \cdots & f_mu_L \end{bmatrix} \times \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_L \end{bmatrix} \tag{8}$$

We then calculate matrix A and the corresponding label vector B from the data. The features are normalized so that all of the feature values fall into range 0 to 1. By feeding the feature matrix and label vector into machine learning model, we can get the weight vector. In addition to using the original business scores from the Yelp dataset, we use the business scores calculated from the HITS-based algorithm as the label vector B as well and compare the weights learnt, which is shown in TableI. We tried both linear regression and support vector regression (SVR) [8] to learn the weight vector. The root mean square error from the latter is a lot smaller, so the weights shown in TableI are learnt from SVR.

Using original business scores as the label vector, useful votes is the most positive feature, followed by average stars and review counts. The idea that useful votes a user receives and her credibility are positively correlated seems quite reasonable. At the first sight, average stars positively correlates with user credibility looks a little weird. However, from Figure *c*, we see that the average stars of most users are from 3.5 to 5. Thus, it is possible that users with high average stars have higher credibility simply because we have more high-average-star user samples. Cool votes turns out to be the most negative feature. Maybe it is because the definition of cool is not that obvious comparing to other kinds of votes. Thus the cool votes number is not so informative.

Using HITS-learnt business scores as the label vector, we see that useful votes and average stars are still important positive features. However, score difference and absolute score difference become important as well, especially the latter. The large negative weight indicates that the bigger the absolute score difference, the lower credibility the user has. This matches the conformity hypothesis, with roots in the social psychology of conformity [2]: a review is evaluated as more helpful when its star rating is closer to the consensus star rating for the product for example, when the number of stars it assigns is close to the average number of stars over all reviews. Thus, if the absolute sum of the difference between the star rating from a user u and the consensus rating is small, the reviews written by this user u might be deemed more helpful and hence the user should receive high credibility score. This effect is much more obvious while using HITS-learnt business scores because the business scores in HITS are weighted based on the score differences (details in the section above). The validity of our HITS-based scoring algorithm is hence preliminarily justified.

VI. Community Detction

A. Model Formulation

For now we already have new scores for each business, but we want personalized score for users. Therefore, we want to detect communities in users and when we give the score of a business for a community, we give a higher weight for the score given by other users in the same community. In this way, different communities will see different scores for a business that are personalized to them.

In this part, we create an undirected, weighted graph that includes some of the users. We calculate Jaccard similarity between two users and if the Jaccard similarity is above a certain threshold, we create an edge between the two users and use the Jaccard similarity as the edge weight.

The Jaccard Similarity between two users $u1$ and $u2$ can be calculated as the

$$JaccardD_{u_1,u_2} = \frac{B_{u_1 \cap u_2}}{B_{u_1 \cup u_2}} \tag{9}$$

where $B_{u_1 \cup u_2}$ is the number of business user $u1$ and $u2$ have both reviewed. And $B_{u_1 \cap u_2}$ is the number of business user $u1$ or $u2$ has reviewed.

In detecting network communities, we will be using a modified version of the Hierarchical Agglomerative Clustering(HAC), repeatedly join together the two communities whose amalgamation produces the largest increase in modularity. We apply some implementation techniques as discussed in [5] and extend to network graph with weighted edges to speed up the clustering process. The detailed algorithm is shown below.

B. Speeded HAC Algorithm for Community Detection

We define A_{vw} be the adjacency matrix of the network thus A_{vw} is the edge weight between vertices v and w if they are connected, here we store A_{vw} as a sparse matrix. We define the modularity Q of the network to be

$$\begin{aligned} Q &= \frac{1}{2m}\sum_{vw}[A_{vw} - \frac{k_v k_w}{2m}]\delta(c_v, c_w) \quad (10)\\ &= \frac{1}{2m}\sum_{vw}[A_{vw} - \frac{k_v k_w}{2m}]\sum_i \delta(c_v, i)\delta(c_w, i) \\ &= \sum_i (e_{ii} - a_i^2) \end{aligned}$$

where

$$k_v = \sum_w A_{vw} \tag{11}$$

$$e_{ij} = \frac{1}{2m}\sum_{vw} A_{vw}\delta(c_v, i)\delta(c_w, i) \tag{12}$$

$$a_i = \frac{1}{2m}\sum_v k_v \delta(c_v, i) \tag{13}$$

HAC involves finding the changes in Q that would result from the amalgamation of each pair of communities, choosing the largest of them and performing the amalgamation. The network can be considered as a multigraph, and each community is represented by one vertex and the edges internal to the vertex are considered to be self-edges. In this way, merging of two communities i and j can be treated as replacing the ith and jth rows and columns by their sum. Noticing for an adjacency matrix, most of the elements would be zeor. Thus some data structures for a sparse matrix would be more efficient. But for the adjacency matrix, finding the pair i, j and calculating ΔQ_{ij} would be very time-consuming.

As in [5], rather than maintaining the network adjacency matrix and calculating ΔQ_{ij}, we would maintain and update a matrix of value of ΔQ_{ij}. Because joining two communities without edges between them would not make an increase in Q, we could only store ΔQ_{ij} for pairs of i, j which are joined with one or more edges. And since this matrix would also be a sparse one, we can use a countermap to store it. In addition, to efficiently get useful data, we used max heap to keep track of largest ΔQ_{ij}. All these improvements would result in a considerable saving of both time and memory.

As described above, the initial state for the cluster is that each vertex is a sole member of a community of one. Then

$$e_{ij} = \begin{cases} 1/2m & \text{if i,j are connected} \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

$$a_i = \frac{k_i}{2m} \tag{15}$$

$$\Delta Q_{ij} = \begin{cases} 1/2m - k_i k_j/(2m)^2 & \text{if i,j are connected} \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

The algorithm now can be defined as below

Algorithm 2 Speeded HAC Algorithm

```
Data: A Network of Users
Result: Detect the community based on the Jaccard Similarity
Initilization
Calculate the initial values of $\Delta Q_{ij}$ according to (16)
Calculate the initial values of $a_i$ according to (15)
Populate the max-heap wiht the largest element of each row of the matrix $\Delta Q$
Set a final cluster number to reach
while True do
    Pop the largest $\Delta Q_{ij}$ from the max Heap H
    Update Δ Q matrix
    Joining communities i and j :
     remove ith row and column
     update jth row and column
    if k is connected to both i and j then
        $Q'_{jk} = \Delta Q_{ik} + \Delta Q_{jk}$
    end
    if k is connected to i but not to j then
        $Q'_{jk} = \Delta Q_{ik} - 2a_j a_k$
    end
    if k is connected to j but not to i then
        $Q'_{jk} = \Delta Q_{jk} - 2a_i a_k$
    end
    Update $a_i$
     $a_i = 0$
     $a'_j = a_j + a_i$
    Update H
     update PriorityQueue followed by each node
     update the max-heap H
    Stopping Criteria
    if the number of clusters has reached a preset number then
        Break
    end
end
```

C. Running Time Analysis

Let's define the degree of node i and node j(the user i and user j) in the reduced graph to be $|i|$ and $|j|$. Updating jth row means to remove ith row and update the values of jth row as the sum of them. Since we store rows as balanced binary trees, each of these $|i|$ action would take $O(\log |j|) \leq O(\log n)$ time. To update all the elements of jth row, it would take at most $O((|i| + |j|) \log n)$.

We also have to update the max-heaps for each row and the overal max-heap H. Reforming the max-heap corresponding to the jth row would take $O(|j|)$ time.Updating the max-heap for kth row would take $O(\log |k|) \leq O(\log n)$. Since we have changed maximum elements on at most $|i| + |j|$ rows, we need to do at most $|i| + |j|$ updates of H, with each time of $O(logn)$ time. Thus the total time for updating max-heap is $O((|i| + |j|) \log n)$.

Finally, for the updates of a_i, it's a constant time operation.

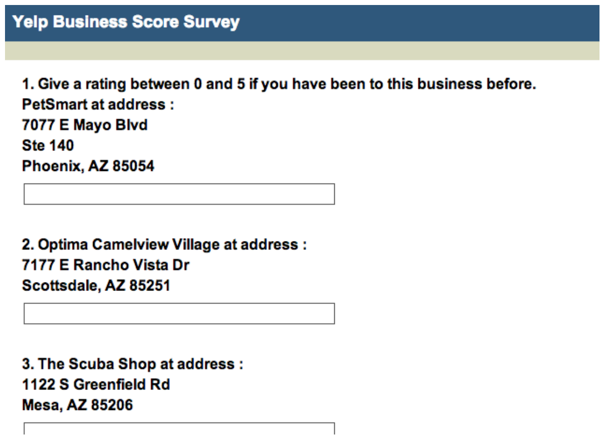
The worst case of all the above updating is that the degree of a community is the sum of the degrees of all the vertices in the original network conributes its degree to all of the commnities it is a part of, along the path in the dendrogram from it to the root. If the dendrogram has depth d, there are at most d nodes in this path, and since the total degree of all the vertices is $2m$, then we would have a running time of $O(md \log n)$.

VII. Results and Evaluation

A. HITS-based new score result

The distribution of the HITS-based new score is shown in Figure a. From the plot, we can see that our algorithm tends to make the scores distribute more uniformly, i.e. less businesses with scores centered at 3.5 4.0, but more extremely high or low scores. One possible reason for this is that the iterative HITS algorithm would amplify the small deviations. The evaluation below also hints this potential problem with the newly calculated scores.

B. HITS-based new score evaluation


Yelp Business Score Survey

1. Give a rating between 0 and 5 if you have been to this business before.
PetSmart at address :
7077 E Mayo Blvd
Ste 140
Phoenix, AZ 85054

2. Optima Camelview Village at address :
7177 E Rancho Vista Dr
Scottsdale, AZ 85251

3. The Scuba Shop at address :
1122 S Greenfield Rd
Mesa, AZ 85206

Fig. 1: Survey.

In order to validate the new scores generated by the HITS algorithm, we designed a survey and found acquaintance to help fill out the survey. We first filter out businesses with the difference between the new score and the old score below a certain threshold t, and then randomly choose 20 businesses to make a survey as shown in Figure 1. A subject is asked to give a rating between 0 and 5 to the business if she/he has been there

TABLE II: Survey Results

Business	Old Score	New Score	User Rating
El Pollo Loco	3	4.9	4
Peter Wong	3	4.6	4.5
Riviera Opticare	4.5	2.7	4
PetSmart	4.5	1.6	3.2
CVS	4	2.1	3
Mid City Kitchen	3.5	1.9	3.8
Wendy's	2	3.6	2.5

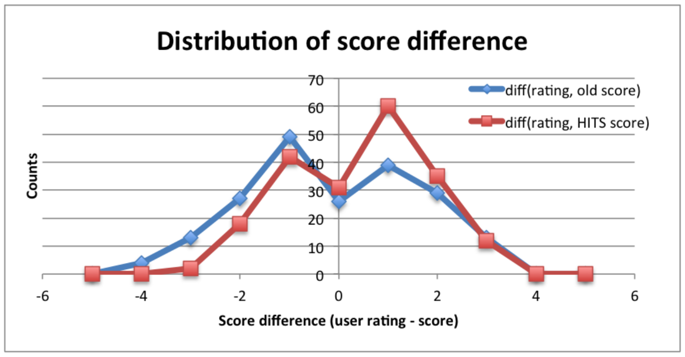


Fig. 2: Score Difference Distribution.

before. We collected 200 ratings from 20 subjects. A few samples are provided in Table II.

Figure 2 shows the distribution of score difference (rounded to the closest integer) between the rating and the original score from the dataset (blue line) and that between the rating and the new score generated from HITS (red line). From the plots, we can see that most score differences center at 1 and -1, i.e. the user rating is usually 1 (0.5 1.5) above or below the old/new score.

In addition, the red line has more pairs with score difference 1 than the blue line does. This is because our HITS system tends to lower the original business score, as shown in Figure a. Thus, the user rating is frequently higher than the score given by us. We also calculated the total score difference across the 200 pairs in order to quantitatively compare the old and the new scores more easily. The total score difference between the ratings and the old scores is 294, i.e. 1.47 per business, and that between the ratings and the new scores is 280, i.e. 1.4. Thus, the new scores from HITS match the user ratings better than the original scores from the subjective evaluation.

C. Community Detection Result

After applying the Jaccard similarity threshold, we created a graph of 17430 nodes that stands for users and 95866 edges that have edge weight of Jaccard similarity between the two nodes. Then we do community detection on those users. As a result of the community

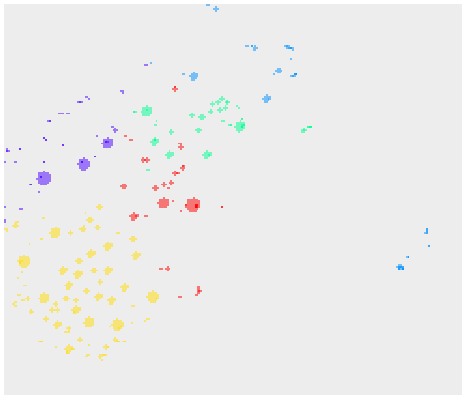


Fig. 3: User Cluster Distribution.

detection algorithm described above, we formed 336 clusters. A plot of the cluster distribution is shown on Figure 3.

D. Community Detection Evaluation

After community detection, we will get a list of all the clusters that contain a certain number of users. For a certain user, we come up with three different ways to get the score of a business that is personalized to the user itself. Suppose U_t is in cluster C_p, business score in yelp of business B_t is BS(B_t), the new score of a business B_t by HITS alogirthm is HITS(B_t), the credibility of a user U_i is $C(U_i)$.

$$Score1(U_t, B_t) = \frac{\sum_{U_i \in C_p}^{n} S(U_i, B_t) + n \cdot BS(B_t)}{2n} \quad (17)$$

$$Score2(U_t, B_t) = \frac{\sum_{U_i \in C_p}^{n} S(U_i, B_t) + n \cdot HITS(B_t)}{2n} \quad (18)$$

$$Score3(U_t, B_t) = \frac{\frac{\sum_{U_i \in C_p}^{n} S(U_i,B_t)C(U_i)}{\sum_{U_i \in C_p}^{n} C(U_i)} + HITS(B_t)}{2} \quad (19)$$

In Score Criterion 1, when we give score of a business for a certain user, we will combine the score that other users give to the same business within the same community with the actual score of the business given by Yelp.

In Score Criterion 2, instead we will combine the score given by other users in the same community and the new score that is generated by our HITS algorithm for the business.

In Score Criterion 3, we will consider user credibility of users when we get scores given by other users in the same community and we will also consider the new score given by HITS algorithm.

In order to evaluate how our new scores are related to the users actual score of a certain business. We get all scores that users give for businesses by traversing all the reviews and we will compare the score that is given by our algorithm to the actual score that a user gives to a business. Notice here that the score we come up with wont depend on the actual score that the user gives. We will see how many percentages of those scores given by our algorithm differ from the actual score that the user gives to the business by less than 0.5, which means how close is the score given by our algorithm to the actual score that the user gives. Here we select all users that are not disjoint, which means they belong to a cluster and for review scores we only choose scores given by particular users and also other users in the same community have reviewed this business before.

TABLE III: Accuracy By Different Methods

Criterion	Score 1	Score 2	Score 3
Test Accuracy%	77.4	78.5	79.4

The test accuracy obtained by the three score criterions is given in Table III. When we apply Score Criterion 1, out of all the 6511 scores as test, 5042 scores given by our personalized algorithm differ from the actual score that users give by less than 0.5. The test accuracy is around 77.4%.

When we apply Score Criterion 2, out of all the 6511 scores, 5109 scores we got differ from the actual score by less than 0.5. The test accuracy becomes 78.5%, which means generally the score given by our HITS algorithm can reflect users actual tastes better. When we apply Score Criterion 3, out of all the 6511 scores as test, 5168 scores that we got differ from the actual score by less than 0.5. We got around 79.4% of scores quite close to the actual score. We can see that by incorporating user credibility, we got a little improvement in test accuracy.

Overall, the score of a business generated by our algorithm is quite close to the actual score that a user gives and we achieve a test accuracy of around 79.4%.

VIII. Conclusion

The goal of this project is to generate a new set of business scores which take into account the user credibility and the community structure. In order to achieve this, we use a HITS-based algorithm to calculate the new business scores and the user credibility scores based on the score difference between the score given to a business by a single user and the consensus business score. The original scores and the HITS generated scores are evaluated subjectively. The evaluation shows that the new scores from HITS match the user ratings better than the original scores. More specifically, the score difference between a rating and an old scores averages to 1.47, and that between a rating and a new score averages to 1.4. However, we also notice from the evaluation result that our HITS algorithm tends to extremify the business scores, i.e. update the original scores with a very low or a very high score, especially the former. This is something we should examine and fix in the future.

Support vector regression is used to learn the important features that decide the credibility of a user. Useful votes and average stars are identified as useful features that positively relate to the credibility score. The absolute sum of the difference between the business rating from a user and the consensus rating is negatively related to the credibility score while using the HITS generated business scores as the correct scores. This matches the conformity hypothesis which says that a review is evaluated as more helpful when its star rating is closer to the consensus star rating for the product.

In detecting network communities, we use a modified version of the Hierarchical Agglomerative Clustering(HAC), repeatedly join together the two communities whose amalgamation produces the largest increase in modularity. We use some additional data structures to speed up the clustering process. The whole clustering can be done in 1 minute. At last, we formed 336 clusters covering 17430 users. A test is done in evaluating the detected communities and we achieve an accuracy of 79.4% on all tested users.

References

[1] Yelp data chanllenge: www.yelp.com datasetchallenge. September 2012.

[2] Rod Bond and Peter B Smith. Culture and conformity: A meta-analysis of studies using asch's (1952b, 1956) line judgment task. *Psychological bulletin*, 119(1):111, 1996.

[3] Allan Borodin, Gareth O Roberts, Jeffrey S Rosenthal, and Panayiotis Tsaparas. Finding authorities and hubs from link structures on the world wide web. In *Proceedings of the 10th international conference on World Wide Web*, pages 415–429. ACM, 2001.

[4] Jinhyung Cho, Kwiseok Kwon, and Yongtae Park. Implicit user credibility extraction for reputation rating mechanism in b2c e-commerce. *International Journal of Intelligent Information and Database Systems*, 1(3):247–263, 2007.

[5] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[6] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[7] Cristian Danescu niculescu mizil, Gueorgi Kossinets, Lillian Lee, and Jon Kleinberg. How opinions are received by online communities: A case study on amazon.com helpfulness votes, 2009.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.