Automated Venue Review System for SFNIGHT

Monica Fernandes | Yisong Chen
SIMS, UC Berkeley
May 7, 2001

Background

SFnight is an information-centric website, focused on venues and their events. Powered by a network of venue owners, promoters, DJs/Bands, SFnight brings the most extensive and intensive nightlife online information and resources to San Francisco nightlife customers. At SFnight website, bar goers will find information about venues and their events in several ways, searching by day, location, venue name, event and venue type, and music style. Bar goers have different nightlife experience in San Francisco. Some might be frequent nightlife consumers, and know a lot about San Francisco nightlife and places; others could be just visitors, with no clue about venues atmosphere and clientele.

Enabled by the power of Internet, SFnight provides multimedia information of venues/events and different ways of search/browsing, to help consumers (bar goers) on their decision on where to go for a night out. To address the issue of information overload that is shown in our user experimental testing, SFnight team members decided to include venue recommendation system. In our initial project design, we defined two kinds of venue recommendations and they are distinct:
- **Editor's Picks.** Editors will select and rate VENUES based on their tastes, their expertise, and on what they think might be relevant for consumers. Editors will pick top venues for different types, for example, for Jazz bar, or for Dancing club.

- **Consumer's Top Venues.** SFnight users would rate venues and write reviews to share their experience with other users. The average rating of each venue will then be shown to all users and serve as a recommendation.

**Problem Statement**

This initial design choice left several problems to be addressed:

- **Users will receive two venue recommendations**, one from editors picks and another from their peers. This may cause confusion because users may not be patient enough to figure out what is going on behind the scene. With the initial design, we found it very difficult to present both recommendations to users without causing confusion. In fact, for most of users who need recommendations, they don’t care that much about how recommendations are generated.

- **Editors were considered the only-experts in the system.** By separately presenting editors’ picks, we assume editors are only expert in our system and their recommendations are valuable. But we have to admit that editors’ decisions are also a matter of tastes and they could make mistake. Our risk of making incorrect recommendation is high. At the same time, some users can be expert at least in two ways. First, frequent SF bar goers can be in better positions to give recommendations than infrequent bar goers; secondly, many users have a profound knowledge of a specific kind of music, and ‘environment’, such as ‘this is
the best bar to Jazz addicts’ or ‘This is where Funk crowd most meet’ and be considered ‘the expert’ by their friends and colleagues.

- **No way to distinguish Good guys and Bad guys.** Good guys are those who tend to give recommendations close to ground truth, while bad guys can be anyone who gives high ratings for his friends’ bars or low ratings based on very specific situation. This is the same problem faced in the peer review system analyzed by Wilensky & Riggs: filtering the quality of reviewers. If the scores of a reviewer is close to the average rating, it is more secure to say that s/he has high ability to judge according to that community of users. We would like to guarantee more relevance to reviewers who provide better judgment, by incorporating reviewer quality into the score. Our initial design can’t distinguish between good guys and bad guys since we simply average reviewers’ ratings.

To address those problems, we decided to develop a recommender system based on the research method proposed by Wilensky & Riggs, in *An Algorithm for Automated Rating of Reviewers*. This method provides a collaborative filtering algorithm and enable us to combine rating of reviewers with item ratings. By incorporating the quality of the reviewer into the system, we can distinguish between good guys and bad guys and our system have better chance to provide ratings closer to ground truth. By using some parameter in the algorithm, we will also be able to treat some users besides our editors as experts. Our editors’ ratings will be combined with users’ ratings since editors’ are considered one of the experts, instead of only expert, in the system. As a result, SFnight users’ will receive only one combined and more accurate recommendation.
Algorithm Implementation

In current implementation, we use the algorithm with two additional factors: $A_i$, $P_i$. Factor $A_i$ is used to leverage reviewers’ experience and $P_i$ is used to give more weight to those items with more reviews. $A_i$ in our system has a little bit different interpretation than $A_i$ in the paper, which will be explained in expert issue.

$$P_i = 1 - \frac{1}{M_i},$$ where $M_i$ is the number of reviews that venue $j$ has received.

The algorithm is implemented in PERL, while all data about clients and their reviews are stored in MYSQL database. We are able to take advantages of relational database to calculate venue ratings and reviewers’ ratings, instead of having to design nice data structures in PERL. We believe if we use a database that support stored procedures, all calculations will be even easier and can be done totally within database.

**Cost solution.** Since we are aware of the cost of calculating venue ratings and reviewers’ ratings, we decided to run the script every ten minutes, instead of running every time when a new review is added. This means when users add a new review, they won’t see the average rating changed right away.

There are couples of important issues we have considered in our implementation:

**Expert issue**

We can identify an expert in two ways. One is from our common knowledge. For example, our editors are supposed to be experts on venues they rate. SFnight can also have some freelancer reviewers who go out a lot and regularly rank venues on our website. Another way to identify an expert is to learn from the
pattern of data. We can recognize experts by matching their preferences with venue types. Integrating these two kinds of experts requires different implementations.

**We know experts outside the system**

Experts can be our internal editors or freelance reviewers. Editors should be able to contribute more to the final rating. Especially when there are not many reviews, editors should be able to balance out those few ratings that may be extreme. When more reviews are added, editors’ contribution should be less. Freelancer reviewers should also be given higher weights in order to contribute more to our system.

We implement this by controlling factor $A_i$. $A_i$ is composed of two parts: the first part is used to compensate experience reviewers and punish inexperience, as the one defined in the paper. Instead of using $A_i = 1 - 1/N_i$, we use $A_i = 1 - 1/(N_i+1)$ because we still want to count the person who has only one review. The other part is the weight $W_i$ we assign to experts. We give editors a high $W_i$: 3 (base $W_i$), general users 1, and freelance reviewers between 1 and 3. So $A_i$ will finally be:

$$A_i = W_i \times (1 - 1/(N_i+1))$$

Editors’ $W_i$ 3 is only a base score, and it will be adjusted by how well their reviews are. In our system, users can rate each review written by an editor. To judge how good an editor is, we will average all ratings users provide for all reviews written by the editor.

Notice that the higher $N_i$ is, the more slowly that $A_i$ increases. We think this is a desirable feature of $A_i$ since as people have more reviews, they should be compensated less for writing one more review.
We know experts by their preferences

Another way to recognize an expert is to look at users’ preferences. In our sign up process, users specify their preferences for types of music and events, and those types match with venue types. This makes it possible for us to recognize what field a user is good at and therefore allow him/her to contribute more to final ratings of those venues in that field. For example, we might consider that a reviewer who likes Jazz might know more about jazz bars than others who don’t like. Consequently his ratings for Jazz bars will be more valuable to other users, whatever they have affinity or not. At the same time, this user should contribute less to the ratings of Dancing Club which s/he is not good at.

To assign different weight for each venue a user rate, we will use the following formulas proposed by Wilensky, to calculate each venue’s rating.

\[ A_i = \frac{\sum_{i \in R_j} W_{ij} R_{ij}}{\sum_{i \in R_j} W_{ij}} \]

in which \( W_{ij} \) is reviewer i’s expertise rating for venue j.

Spoofers issue

There could be different kinds of spoofers and each has different behavior patterns. In our system we consider typical spoofers as those who intentionally give high ratings for his venues or his friends’ venues, and give low ratings for his competitors’ venues. We will detect them by noticing that a venue receives a bunch of ratings greatly deviated from our editor rating, from users who mostly newly registered. This detection will at least cover two situations:

- Venue owners or promoters create different logins to rate his own venue.
- Venue owners or promoters send his friends to SFnight to give high rating for his venues. Since their friends are probably not registered in
SFNIGHT, and some of them will have to register first to give a ranking, we can detect them by seeing a bunch of newly registered users give high ranking to a venue.

Another kind of spoofer will be those who intentionally rate many items at their known average to obtain a high reviewer rating. We can detect these spoofers by noticing s/he writes a lot of reviews in a short period of time and most of its rating are close to average rating, which is too good to be true.

Once a spoofer is detected, we simply set their $N_i = 0$ and their ratings will be out of our calculation.

**Interface Implementation**

**Rating for a venue**

On venue page, user will be able to see the average rating for the venue and the list of reviews and ratings contributed by their peers. The user rating ranges from 1 to 5, while average ranking for venue scale from 1 to 100. The two main reasons for this different scale, in spite of the fact that users may be confused:

- When users' ratings are integers, the calculated ratings usually are not. If we round the result rating into 1 to 5 scales, we lose a lot of information

- Since the average ratings script runs every 10 minutes, when users add a new review, they won’t see the venue's average rating change right away. If we use 1-5 scales for average ratings, users will be very aware of that.

To prevent users' confusion, we will put a link besides the average rating and explain how we calculate the average ratings.
Rating for a venue/event description

Right after each venue/event description, there is a rating for users to rank how helpful the description is. A good Editor would be the one who provide a description that can help users to make their decisions on where to go for outing. If users approve editor’s description, they might also be confident about his/her ratings. By having a good score in several venues and events descriptions, it is more likely editors will give reasonable rate to venues. Rating Editors also will help us in evaluating how different editors’ reviews are from players’ description for their own venues/events, and provide a metric system for us to control the quality of editor reviews.

Our main concern for this interface design is that it has to be super easy, otherwise users won’t bother to do it. That is why we design in such way that ranking a description is just as simple as one-click.

Other pages

One-rating-for-one-venue strategy also improves a lot of other pages on SFNIGHT and at the same time make designer’s work much easier. For example, on homepage we no longer have to present editor picks top venues and top 10 venues from users’ ranking. This improved interface is much clearer to our users. Another example is: when users search for venues, venues will be listed by one venue rating, instead of being listed by users average ratings with a mark indicating whether this is an editor’s pick.
Discussion

Besides being able to incorporate experts into the system and detect spoofers, the algorithm also bring some other benefits to our system:

- **Quality control.** Since our editors’ reviews are rated by users and their ratings are compared to users’, editors are more likely to give more objective evaluations. SFnight would also have a way to evaluate these official ‘reviewers’ (Editor’s) and compare them with other reviews written by information providers enabled by our SFnight website (promoters and bar owners). The system will make it easier for us to recruit freelance reviewers too since their ratings can be monitored.

- **Venue environment may change overtime, and old reviews shouldn’t be punished.** Venues changes ownership, management, staff, and consequently might change their service quality, the atmosphere and clientele. They might improve, be the same or decline. We a venue change its environment, old reviewers will be punished since their old rating may deviate from the average although old ratings did reflect the real status at that time. We can address this issue by setting a timestamp for each venue. Only reviews after that time will be counted toward venue ratings.

- **Ways of building and leveraging recommendation communities.** At our initial design, we have not considered the idea of making recommendations a more interactive system, and providing users with ways to find others that have similar affinities. In our future development, we will include “view user profile”, by which users can view a reviewer’s personal profile to identify reviewer’s preferences, favorite venues or DJs, etc.
- **Balancing low rating participation.** The combination of editors' rating and users ratings will sort of balance out the potential bad influences extreme ratings can cause.

- **Limited set of recommenders leverage subjectivity.** Consumers usually like to interact with friends and acquaintances, to share experiences and get recommendations. And recommendations are based on subjectivity. The larger the set of recommenders, more objectivity the rating system gains.

\[
\sum_{i \in R_j} \frac{W_i R_{ij}}{W_i}
\]