New York University

Introduction to Data Science Project Report

Beer Recommender System

Qingxian Lai

Peter Li

Jia Xiao

Yijun Xiao

Instructor:

Prof. Brian d'Alessandro

December 19, 2014

Contents

1 Business Understanding

BeerAdvocate.com is an online beer-rating site founded in 1996 by brothers Todd and Jason Alstrom. The site centers around beer reviews provided by registered enthusiasts and industry professionals. Users have the option of providing a numerical score and text review for any beer in the system. In addition, users can browse reviews using the web site's search feature.

BeerAdvocate generates most of its revenue through online advertisement. With more than 45 million page views/month and 3 million unique visitors/month, BeerAdvocate is ranked 1,620 in the United States by web traffic [1][2]. As in any businesses analysis, two questions immediately come to mind. First, is there any ways BeerAdvocate can deliver additional value to users? And second, is there anything more BeerAdvocate can do to increase revenue?

To answer these questions, it is helpful to consider what might negatively affect the user experience. For beer enthusiasts, BeerAdvocate is a great website for logging reviews. However, BeerAdvocate should not only function as a repository for reviews. Without an effective way to interact with the reviews, user may become frustrated as they manually sort through reviews to find new beers they might enjoy.

For new users, the problem is worse. Not only do they have to read reviews in a random fashion, but they often times lack the domain expertise of long-time users. The diversity and tremendous amount of beers can make this task seem impossible. Even with the Top 250 beer provided by the website, users typically browse only the top dozen.

We believe these problems are not well addressed by BeerAdvocate. Even though the site collects a great amount of data regarding different users opinions, it does not utilize the data effectively. Instead, they treats the information passively, putting them down like different blocks for users to browse and pick up. By building systems to effectively leverage existing data, we can provide a much more convenient way for users to explore different aspects of beer on this website.

For our project, we propose deploying a beer recommendation system on BeerAdvocate that will increase customer engagement and ultimately revenue. The main goal of our project is to use the information given in each beer rating to build a recommendation system that recommends beer that is more likely to be accepted by a user. By adding a recommendation system, users don't have to waste time reading though myriad reviews. Instead, they can start by looking at beers they probably like, and then explore others.

2 Data Understanding

For our data mining process, we need the access to the dataset containing users rating for each beer. Such a dataset can be found from the Stanford web data library (`http://snap.stanford.edu/data/web-BeerAdvocate.html`)[1]. The data we downloaded is of the text format as shown in Figure 1.

As we can see besides the information about the beers name, brewery, ABV etc., each registered user gives a rating from 1 to 5 with respect to look, smell, taste, feel and overall.

[1]The data has been removed at the request of BeerAdvocate at the time of writing.

```
beer/style: American Amber / Red Ale
review/appearance: 4
review/aroma: 4
review/palate: 4
review/taste: 4
review/overall: 4
review/time: 1171415576
review/profileName: flexabull
review/text: Beer pours a dark clear amber color with a nice sturdy
head. Aromas are of fresh pine like hops. Not overpowering, but what
is there is nice. Flavors are similar to the Ashland Amber, but
```

Figure 1: Typical review format

Then they write a text comment related to the characteristics they think best describe the beer. The data is organized row-wised and delimited by newlines. Each line has a key/value pair. The data has a time span from Jan 1998 to Nov 2011. It has the record of 33,387 users and 66,051 beers, and the total number of review summed up to be 1,586,259.

We found out a lot of users have only rated very few beers and a lot of beers have only been reviewed one or two times. If we sort the beer according to how many times they have been reviewed, we can acquire a plot with x-axis be the number of reviews, and y-axis the count of corresponding beers. Then we can see from Figure 2 exhibit a very long tail, which means a small portion of the beers consumes most of the reviews.

Similar situation happens to the user. If we count the users according to how many reviews they have written, we get Figure 3.

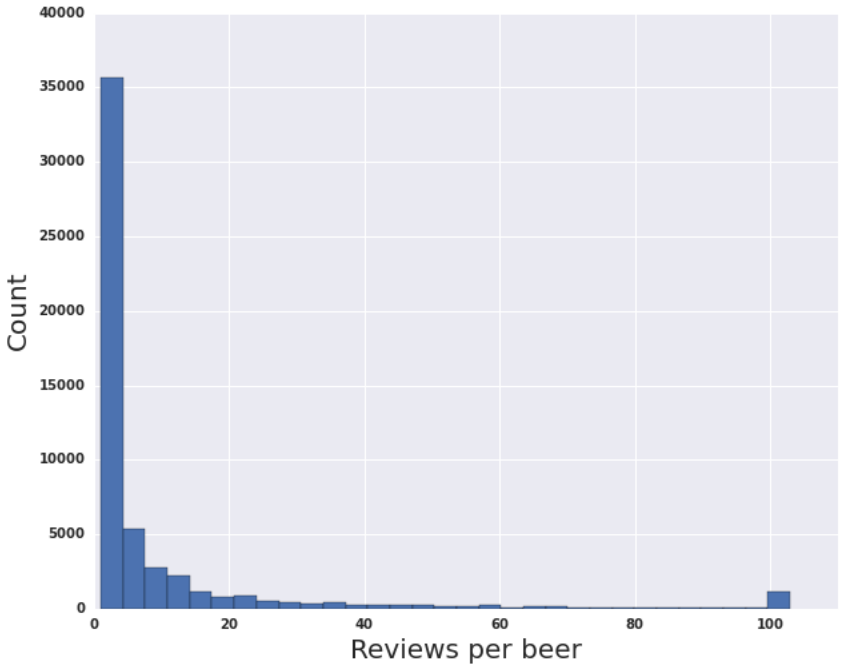

Figure 2: Histogram of beer reviews

3 Data Preparation

Raw review data was in a format that cannot be fed into training directly. We need to prepare the data into better format.

We first substituted all tabs (\t) to spaces (\s) in our dataset. Then converted multi-line review into one line for each review separated by tabs (\t). We didn't choose csv file or txt file because of the frequent occurrences of commas and spaces.

Out of all five numerical ratings, we chose overall ratings to represent users' preferences. Thus the final data format is a csv file containing user-beer-rating triplets.

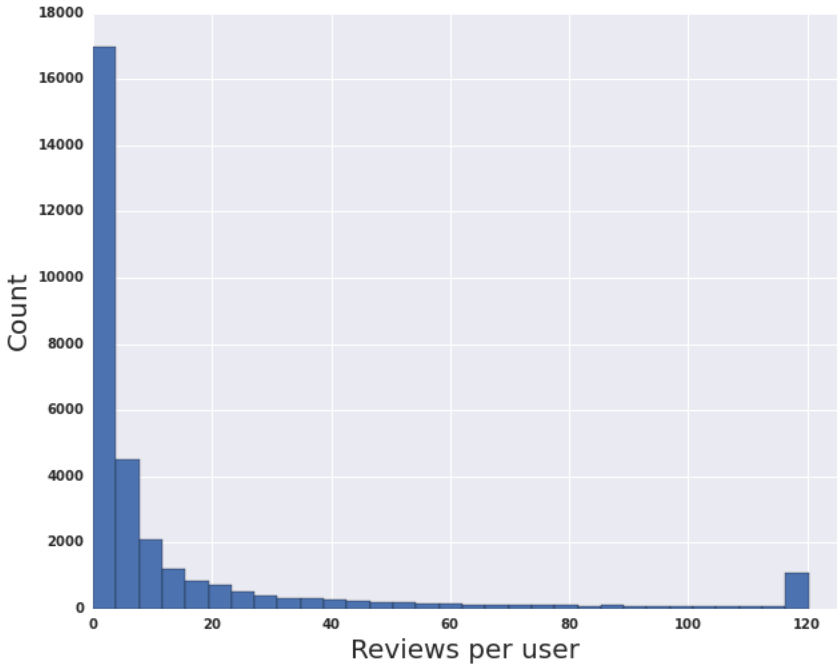

Figure 3: Histogram of user reviews

4 Modeling

We developed two separate sub-systems complementing each other to form a more comprehensive recommendation system. For registered users that have reviewed beers more than a preset threshold, we used collaborative filtering algorithm to predict ratings for those users and recommend top rated beers. For new users and unregistered users, because of the cold start problem from which collaborative filtering algorithms typically suffer, we provided a non-personalized model based on clustering similar commented beers.

4.1 Collaborative Filtering Models

4.1.1 Preprocessing

Aside from basic data cleaning described in Section 3, we need additional processing steps to prepare the data for collaborative filtering training. More specifically, we need either a user-beer rating matrix or a list of user-beer-rating triplets. For medium to large scale recommender systems, we decided to use user-beer-rating triplets because of the size and sparsity of the demanded matrix.

We also decided to drop users and beers that have less than 20 reviews considering the nature of collaborative filtering algorithms. After filtering out unqualified (review-quantity-wise) users and beers, we got $1,296,590$ out of the original $1,586,613$ reviews.

Out of all five numerical ratings each users gave, we only used overall ratings because they best represent users' preferences.

4.1.2 Methods

We tried three different kinds of collaborative filtering algorithms: item similarity, matrix factorization, and popularity based models.

Item similarity model is a neighborhood-based method for collaborative filtering. It first computes the similarities between items, in our case beers, by observing users who reviewed both beers. The unknown ratings are then predicted by averaging similar beer ratings for specified user. Recommendations are made based on top promising beers according to our predictions.

In item similarity models, different similarity measures have different performances based on the property of data. Most commonly used similarity metrics are Jaccard similarity, Cosine similarity, and Pearson correlation. A closely related method is user similarity method. Rather than finding similar beers, it measures similarities between user pairs and predict ratings based on similar users' ratings.

Neighborhood methods enjoy considerable popularity due to their simplicity, efficiency [3]. But in practice, a large number of ratings from similar users or similar items are not available, due to the sparsity inherent to rating data. Consequently, prediction quality can be poor [4].

Matrix factorization model assumes each beer has its own set of latent properties and users have their own preferences for each latent property. The ratings can be represented as interactions between these latent properties and preferences. More formally, the rating of user i on beer j can be formulated as

$$R(i,j) = \mu + a^T x_i + b^T y_j + u_i^T v_j,$$

where μ is a global bias; x_i, y_j are feature vectors for user i and beer j respectively and a, b are their corresponding weights; u_i, v_j are the latent preference vector for user i and latent property vector for beer j.

Matrix factorization methods combine good scalability with predictive accuracy. In addition, they offer flexibility for modeling various real-life situations [5].

Popularity model is a simple model that recommend most reviewed beers to all users. The recommendations are the same across users, there is no personalizations using this model. Because of its simplicity, it's often used as a baseline model.

GraphLab has a python package implemented a collection of machine learning algorithms. We used recommender module to create all three models. For more information about Graphlab, reader can visit `graphlab.com`.

4.1.3 Model Selection

To select best models from each of the three different kinds of approaches, we performed conventional cross validation techniques.

Full data was splitted into training, validation and test sets. We used training and validation sets to choose hyper-parameters. For item similarity models, the choices of similarity metrics and number of neighbors affect model performance greatly; for matrix factorization models, we need to specify the regularization term. Results of the cross validation are presented in Figure 4 and Figure 5.

According to our cross validation results, we chose Pearson correlation as the similarity metric for item similarity model with number of neighbors set to 500; regularization term for matrix factorization model was set to 1e-5.

Due to different predicting behaviors of these algorithms, simply comparing RMSEs across three methods does not justify the superiority of the one gives lowest RMSE. Thus we need another way to decide our best candidate for deployment. We will discuss this further

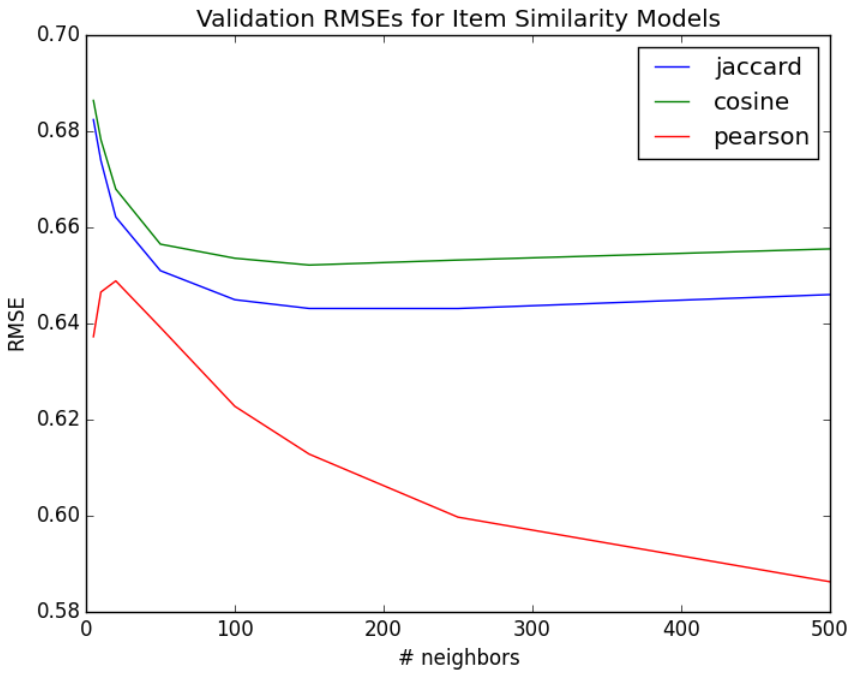

Figure 4: Model selection for item similarity models

in Section 5.

4.2 Grouping Beer Based on Review Text

Collaborative filtering techniques offer a way to generate recommendations for users who have provided a sizable number of past reviews. By analyzing reviews across all users, collaborative filtering techniques learn relationships between users/beers and provides highly customized recommendations based off of learned preferences. However, these systems have a reliance on user review data and therefore suffer from a cold start problem. How can you make recommendations to a user who has not reviewed anything before? Without a user's preferences, it becomes difficult for these systems to make accurate recommendations.

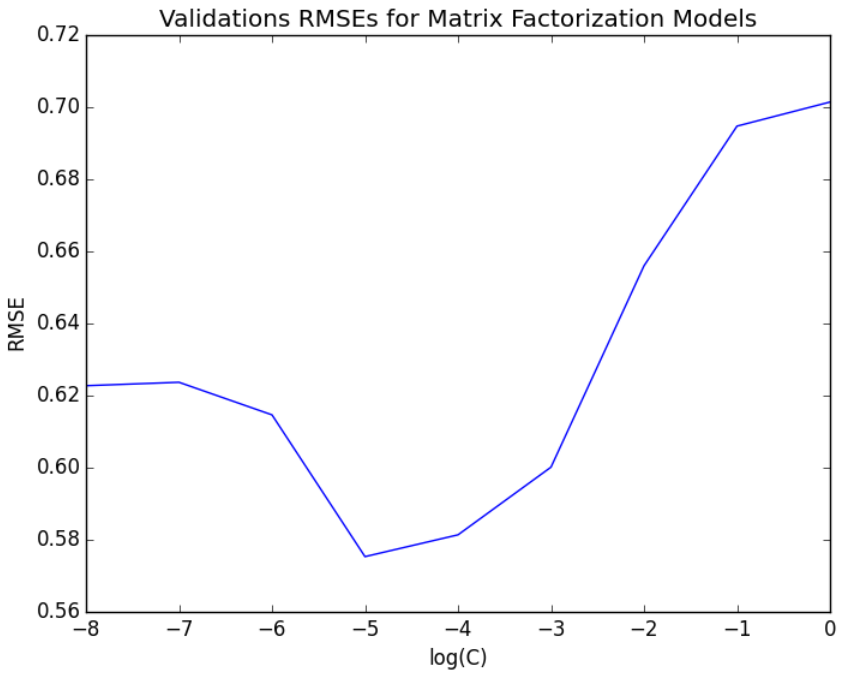

Figure 5: Model selection for matrix factorization

Therefore, we propose a secondary system for users with limited review history.

The problem with limited review history is the difficulty in learning user preferences (without looking at outside data such as demographics). Without user preferences, it is difficult to make a recommendations. To get some sense of user preference, our second system will ask the a simple question:

What's a beer that you like?

Using the response as a proxy for user preference, we fit a clustering model to find similar beers and recommend the ones with higher average reviews. The process can roughly be divided into two sections which we will discuss below.

4.2.1 Data Processing

To fit a clustering model, we first needed features to describe each beers. Given the data, there were a number of features we considered:

1. Beer Style
2. Brewery
3. ABV
4. Review Score (Overall, Palate, Aroma, Appearance, Taste)
5. Review Text

In deciding which features to use, we first considered the question we were trying to answer. What does it mean for two beers to be similar? In our opinion, similar beers are ones that are described in similar ways. Therefore, we focused our efforts on understanding the review text.

The schematic of our feature engineering process is illustrated in Figure 6.

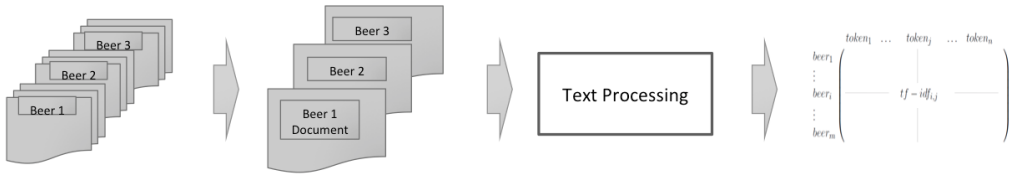

Figure 6: Feature engineering process

First, our cleaned data was filtered to exclude beers with fewer than five hundred reviews. This was done to remove less popular beers and also to make sure we had enough review text to work with. Using a cutoff of n = 500, we were left with 590 unique beers and roughly

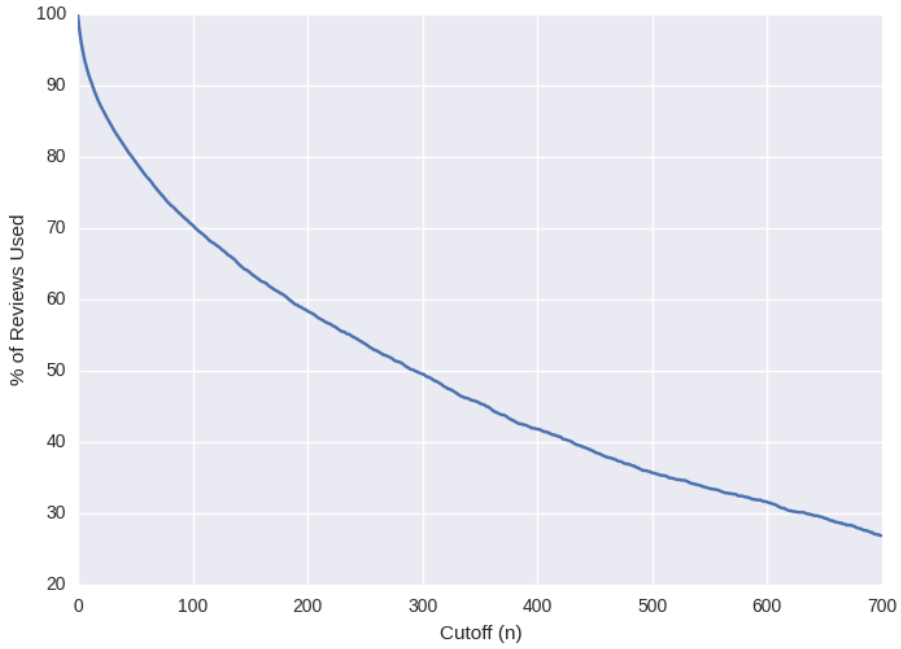

Figure 7: Cutoff rates vs percentage of data used

560 K reviews. Figure 7 shows the trade off between various cutoff rates and the percentage of data used.

After the data was filtered, we grouped the review text by beer type. All the reviews of a beer were grouped into what we called a beer document. Our corpus was the set of beer documents for all the beers in our sample.

With corpus in hand, we turned to feature extraction. First we split each document into separate strings using the Porter Stemmer [6][7]. Next we dropped all the strings that correspond to commonly occurring words that carry no content i.e., stop words. Then the data was passed to the TF.IDF vectorizer in Scikit-learn [8]. This functions turns a collection of stems into a feature vector of TF.IDF scores.

TF.IDF (Term Frequency times Inverse Document Frequency) is a formal metric of how concentrated into relatively few documents the occurrences of a given word is [9]. As a feature, words with high TF.IDF scores are often words that characterize a document well. For example, Aprihop and #9 are both beers that are brewed with apricot puree. Apricot occurs frequently in the beer documents for these two beers but not in other beers, so it should be a good descriptor of these two beers.

The formal definition of TF.IDF is

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}},$$

$$IDF_i = \log_2(N/n_i).$$

Here the term frequency TF_ij is the frequency of term i in beer document j, normalized by the maximum occurrence of any term in beer document j.

The Inverse Document Frequency of term i is the log of total beer documents divided by the number of beer documents in which term i occurs.

Finally, TF.IDF is defined as $TF(1+IDF)$. Note: this is the scikit-learn implementation to include words that have $IDF = 0$ i.e., words that occur in all documents. Typically $TF.IDF = TF \times IDF$. Considering the apricot example again, Apricot has a high IDF since it is rare across all documents and a high TF in Aprihop and #9. Therefore, apricot will have a high TF.IDF for these two beers.

These scores are stored in a feature matrix where each row corresponds to a beer and

each column is a term.

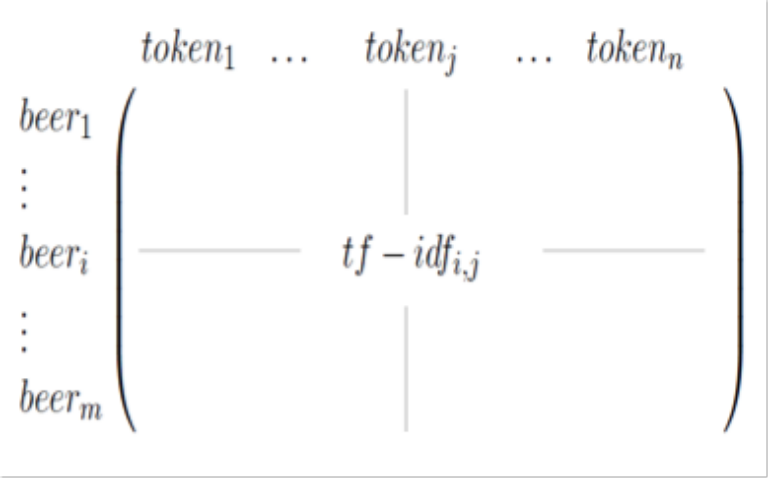

Figure 8: Feature matrix

4.2.2 Data Modeling

After we engineered a feature matrix, we found groups of similar beers via k-means clustering [10]. This is an algorithm that, given a number of clusters k, will group beers into k clusters in such a way that minimized within cluster variation.

To find k, we relied on previous domain knowledge. For small values of k, the clusters were too large and uninformative. Large values of k resulted in many single beer clusters which were equally uninformative for making recommendations. With k =50, we struck a balance between large, general clusters and small, overly-specific clusters.

4.2.3 Recommendation

Finally, to make a recommendation we take the user supplied beer, find its cluster, and recommend the top five beers ranked by average review.

5 Evaluation

5.1 Business Evaluation

One big question we faced is whether these models will eventually improve customer engagement on BeerAdvocate.com. To solve this problem, we build up some A/B tests for the models above, separately. Under the web design circumstance, the goal of A/B testing is to identify changes to web pages that increase or maximize the web usage. In our project, we estimated the web usage by monitoring some key performance indicators [11].

Bounce Rate Single page view visits divided by entry pages, which represents the percentage of visitors who enter the site and "bounce" (leave the site) rather than continue viewing other pages within the same site.

Conversion Rate is the proportion of users who make a conversion. Conversion means a visitor completing a target action. In our case, the action may refer to register as a BeerAdvocate.com user or write a review to a beer he likes.

Daily Page Views means the daily amount of page views of a specific visitor. Page views is the number of times a page was viewed.

Daily Time on Site is the amount of time an user spend on BeerAdvocate.com per day.

The first two indicators are useful to evaluate the content-based model, which is de-

Table 1: Content-based recommendation system A/B test

	A	**B**
Bounce Rate	0.5	0.3
Conversion	0.2	0.4

signed to deal with cold start problem. And the last two clearly indicate the engagement of registered users, so we use them to evaluate the CF models.

Next, we will build up A/B tests based on these indicators.

5.1.1 Experiments on the Content-based Recommendation Model

We can use the following methods to implementing the test.

New users from other websites will randomly see two different landing pages. The control page (Variation A) randomly recommend 5 beers, while the treatment page (Variation B) will guide the users to select their favorite beer, and then recommend 5 beers based on the model. Then we observe their following actions: leave the site immediately (bounce), view some pages then leave, register to be a BeerAdvocate.com user. With this information, we can calculate the bounce rate and conversion rate. Table 1 is an example.

Comparing the difference between the variation A and B with a certain significant level (usually chosen from 95% to 97.5%), the test will draw a conclusion of whether this recommendation system improved the customer engagement.

Furthermore, we can add buttons under the 5 beer recommended by the system which allow users to give feedback about whether they liked the beers or not. This will give us

Table 2: Collaborative filter recommendation system A/B test

	Control	**Treatment 1** Item similarity	**Treatment 2** Factorization	**Treatment 3** Popularity
Average Daily Page Views	5	8	10	6
Average Daily Time on Site	20min	34min	40min	41min

a clear view about the accuracy of the recommendation model. In next part, we will show some methodology for evaluating the collaborative filter model.

5.1.2 Experiments on the Collaborative Filltering Model

The experiment designed are very similar with the one in last part. But this time, we have three different models. Although the factorization model has the least RMSE, experiments are needed to test their ability to improve web usage. Here gives a typical experiment process.

The collaborative filter recommendation system are used to recommend beers to the existing users. When they log in, the system will recommend beers for them. The "control" page will randomly show 5 beers; "treatment 1" page will recommend 5 beers based on item similarity CF model; "treatment 2" page will recommend 5 beers based on factorization CF model; and "treatment 3" page will show 5 beers recommended from popularity CF model. These four pages will randomly present with equal probability. With some web analytic tools, we will know "Daily Page views" and "Daily Time on Site" of these users. Then we can calculate the average value with these data. Table 2 is an example:

The one with the lowest "Average Daily Page View" and highest "Average Daily Time on Site" is the best variation. Comparing the four variations at the same time will generate some confusion. For instance, the one with the lowest "Average Daily Page View" may not be the one with the highest "Average Daily Time on Site". To achieve a more precise conclusion, we need to build up A/B test between each pairs.

6 Deployment

6.1 Deployment of Recommendation System

6.1.1 Content-based Recommendation System

For new users, we will use the content based recommendation system to recommend beers for them. Our team built up a web application to give user a user-friendly interface. When the application starts, user will see a window shown in Figure 9.

The user can select a beer he likes in the drop down list. Then our system will display a list of similar beers along with their style and ABV which is shown in Figure 10.

BeerAdvocate.com can insert this widget into his homepage to deploy it. They can connect the beers to the pages showing detailed beer information, or connect to a register page to encourage the visitor registering.

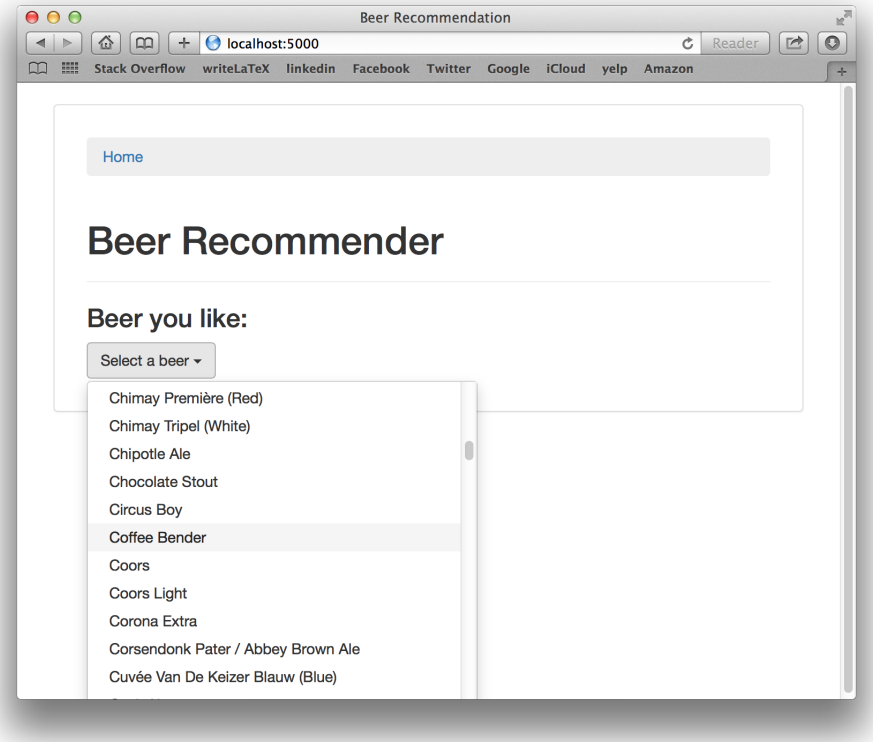

Figure 9: Beer Selection Page

6.1.2 Collaborative Filter Recommendation System

For BeerAdvocate.com users, we can use the CF model to recommend beers to them when they log in the system or when they browse the site. One advantage of this model is that, it does not need to be trained frequently. The website engineers only need to retrain the model at the beginning of every month.

6.2 Ethical Consideration and Risk

One moral consideration is: Is it ethical to recommend beers? Under the optimal situation, the recommendation works very well. As a result, users of BeerAdvocate.com may drink

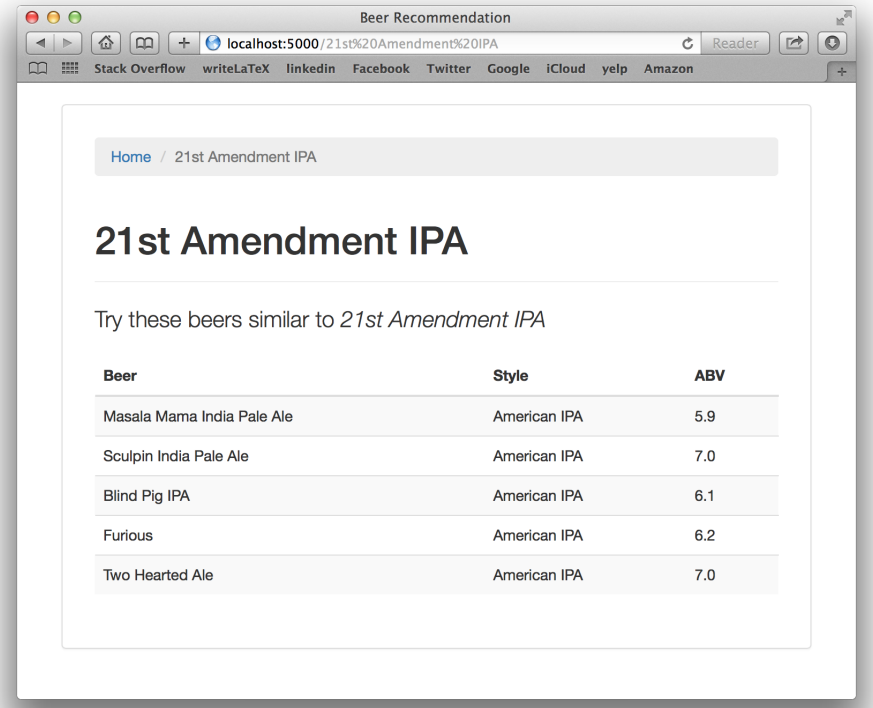

Figure 10: Beer Recommendation Page

more beers than before. As we all know, beer is an alcohol drink. Drinking too much beer will lead to some serious problems, e.g. it will impair people's abilities to drive; the drunk people may hurt their families or even strangers. To address these problems, we can add some warnings besides beer recommendations, showing the bad results of getting drunk.

7 Appendix

All group members contributed their best efforts to this project. We listed here each member's contributions on the writing.

Qingxian Lai Section 5 Evaluation, Section 6 Deployment.

Peter Li Section 1 Business Understanding, Section 4.2 Grouping Beer Based on Review Text.

Jia Xiao Section 1 Business Understanding, Section 2 Data Understanding, Section 3 Data Preparation.

Yijun Xiao Section 3 Data Preparation, Section 4.1 Collaborative Filtering Models.

8 References

1. `http://www.beeradvocate.com/about`.

2. `http://www.alexa.com/siteinfo/beeradvocate.com`.

3. Ricci, F., Rokach, L. & Shapira I.B. *Recommender Systems Handbook*, Springer, 2011.

4. Wang, J., de Vries, A.P. & Reinders, M.J.T. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. *SIGIR 06 Proceedings of the 29th Annual International ACM* p.501-508, 2006.

5. Koren, Y., Bell, R. & Volinsky, C. Matrix Factorization Techniques for Recommendation Systems. *Computer*, v.42 n.8, p.30-37, 2009.

6. Porter, M.F. An algorithm for suffix stripping. *Program*, 14:130-137, 1980.

7. Bird, S., Loper, E. & Klein E. *Natural Language Processing with Python*, O'Reilly Media Inc., 2009.

8. Pedregosa et al. Scikit-learn: Machine Learning in Python, *JMLR* 12, p.2825-2830, 2011.

9. Leskovec, J., Rajaraman, A. & Ullman, J.D. *Mining of Massive Datasets*. Cambridge University Press, 2011.

10. Lloyd, S.P. Least squares quantization in PCM, *IEEE Transactions on Information Theory* 28(2):129-137, 1982.

11. Burby, J., Brown, A. & WAA Standards Committee. *Web Analytics Definitions*. Washington DC: Web Analytics Association, 2007.