Recommender Systems for a Restaurant & Consumer Dataset

Li-Yen Hsu
November 18, 2017

1. Definition

1.1. Project Overview

Recommender systems are one of the most popular machine learning algorithms for internet business in a variety of areas. By predicting the preference or ratings that the consumers would give to certain products, recommender systems perform efficient personalized marketing, matching the products to their prospective customers [1]. Many companies with recommender systems, such as Amazon, Netflix, and Spotify, have successfully boosted their profit by offering their consumers relevant items at the right time.

The dataset for this project is from the UCI Machine Learning Repository [2], which was originally obtained from a recommender system prototype [3]. These data contain the features of hundreds of restaurants and consumers in Mexico, as well as the ratings given by the consumers to some of the restaurants. The goal of this project is to build a recommender system to predict the ratings that would be given by each consumer to the restaurants he/she has not rated. A list of restaurants with the highest predicted ratings can then be recommended to each consumer.

1.2. Problem Statement

Because consumer ratings are numerical, predicting their values can be treated as a regression task. Using classification techniques is also reasonable since ratings are usually discrete integers. However, a classification approach will likely predict too many ties and therefore prevent us from ranking the restaurants for a consumer. Rather than predicting the exact values of ratings that a consumer would give to certain items, what is more important for a recommender system is to accurately predict the ranking. I therefore attempted to predict continuous values for the ratings in this project.

1.3. Metrics

I used two evaluation metrics to quantify the model performances. The first one is root-mean-squre error (RMSE), which is very standard for regression problems because it indicates how close

the predictions are to the labels. Given m pairs of predicted ratings ($\hat{y}_i$) and true ratings (y_i), RMSE is defined as

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{m}(\hat{y}_i - y_i)^2}{m}} \tag{1}$$

However, for recommender systems, the most important information is the ranking of the items, which RMSE does not take into account. For example, a perfectly ranked prediction can have a terrible RMSE by predicting the ratings at the wrong scale. Thus, the second metric I used is fraction of concordant pairs (FCP) [4]. The concept of FCP is the following. Suppose a user has rated n items, there would be $n(n-1)/2$ unique pairs of ratings. If item A receives a higher rating than item B from this user and the model predicts the same, A and B are a concordant pair (CP), otherwise a discordant pair (DP). FCP is simply the fraction of concordant pairs among all the pairs (summed over all the users). Note that because the ratings are discrete integers, many rating pairs actually have the same true values. These ties are excluded from the calculation of FCP, so FCP is defined as

$$\text{FCP} = \frac{n_{\text{CP}}}{n_{\text{CP}} + n_{\text{DP}}} \tag{2}$$

For a task of top-n recommendations, mean average precision (MAP) is a well suited metric [5] because it evaluates how relevant the predicted top-n list is. Unfortunately, the amount of data in this project is small, as most of the users rated less than 10 restaurants. After splitting the exiting ratings into training/test sets, there are at most five ratings for a user in the test set, and many pairs are ties. Thus, FCP fits this project better than MAP.

2. Analysis

2.1. Data Exploration

The data are nine csv files, including five for the restaurant information, three for the consumer information, and one for the ratings. Three ratings (overall rating, food rating, and service rating) with values of 0, 1, or 2 are given for a restaurant-consumer pair. The three types of ratings highly correlate with each other, with the correlation coefficients ~ 0.65 (food v.s. service), 0.69 (overall v.s. service), and 0.71 (overall v.s. food). Each restaurant in the data is named by a **placeID**[1], and each consumer has a **userID**. There are a total of 1161×3 ratings from 138 consumers for 130 restaurants, so the rating fraction of all the possible restaurant-consumer pairs is only about 6.5%. The content of the nine csv files is summarized as follows.

1. chefmozaccepts.csv: two attributes – placeID and Rpayment.

[1]Throughout this report, I will use bold text for the notation of variables used in my code.

2. chefmozcuisine.csv: two attributes – placeID and Rcuisine.
3. chefmozhours4.csv: three attributes – placeID, hours and days.
4. chefmozparking.csv: two attributes – placeID and parking_lot.
5. geoplaces2.csv: 21 attributes – placeID, latitude, longitude, the_geom_meter, name, address, city, state, country, fax, zip, alcohol, smoking_area, dress_code, accessibility, price, url, Rambience, franchise, area, and other_services.
6. usercuisine.csv: two attributes – userID and Rcuisine.
7. userpayment.csv: two attributes – userID and Upayment.
8. userprofile.csv: 19 attributes – userID, latitude, longitude, the_geom_meter, smoker, drink_level, dress_preference, ambience, transport, marital_status, hijos, birth_year, interest, personality, religion, activity, color, weight, budget, and height.
9. rating_final.csv: five attributes – userID, placeID, rating, food_rating, service_rating.

Some of the less obvious attributes (all categorical) are explained as fellows:

- **Rpayment**: the payment methods accepted by a restaurant.
- **Rcuisine**: the types of cuisine offered by a restaurant or the interests of a consumer.
- **accessibility**: "no accessibility", "completely", or "partially"; accessibility for people with disabilities.
- **Rambience**: "familiar" or "quiet".
- **franchise**: "true" or "false"; whether a restaurant is a franchise.
- **area**: "open" or "closed".
- **other_services**: "one", "internet", or "variety".
- **Upayment**: the payment methods a consumer has.
- **ambience**: "family", "friends", or "solitary".
- **transport**: the transportation method a consumer uses.
- **hijos**: "independent", "kids", or "dependent".
- **activity**: "student", "professional", "unemployed", or "working-class".

Intuitively, the main factors affecting the restaurant ratings are (1) how delicious the food is and (2) how satisfied a customer was with the service provided. These two are definitely not something in the data. However, the features of the restaurants, such as parking options, price, or whether alcohol is available, may still have small influences on the ratings.

2.2. Exploratory Visualization

In order to see how the restaurant features affect the ratings, I first computed the mean rating for each restaurant. Then, for each feature, I can inspect how the mean rating changes with different feature values. To do this, I grouped the restaurants based on the different feature values and then averaged the mean ratings mentioned above for each group. Notice that what I did is NOT splitting individual ratings into groups based on the feature values. Instead, it is first obtaining the mean rating for each restaurant and then grouping. If the former approach were used, the mean rating of a group would be biased towards the restaurants with more ratings. This should be avoided because we want to treat every restaurant equally. Figure 1 shows the distribution of the mean ratings for each of the three ratings.

For a content-based recommender system, finding the relevant features is important. As I will discuss in detail in Section 2.3, what my models learn and predict is the variation of ratings as a function of consumers for a given restaurant. As a consequence, a feature is relevant if different consumers react differently to the change of its value. In contrast, a feature is useless if the change of its value affects the rating equally for all the consumers. In our dataset, the restaurant features that agree with this definition of "being relevant" are **alcohol**, **parking_lot**, **price**, and **smoking_area**. These four features also have their counterparts in the consumer data, which are **drink_level**, **transport**, **budget**, and **smoker**, respectively.

In Figure 2, I show the dependence of mean ratings on the value of **alcohol**; Figure 3 further splits the plots in Figure 2 into different groups of consumers using the categories of **drink_level**, "abstemious", "casual drinker", and "social drinker". Same exercises for **parking_lot** & **transport**, **price** & **budget**, and **smoking_area** & **smoker** are shown in Figures 4 through 9. For **dress_code** and **dress_preference**, I did not find significant difference between different groups of consumers. There are other restaurant features without counterparts in the consumer data, such as **accessibility**, **area**, and **other_services**. As a consequence, I cannot split the data into different groups of consumers. However, one would intuitively think these are irrelevant features because the change of their values either affects the consumers quite equally (e.g., **area** and **other_services**) or simply has tiny influence (e.g., **accessibility**). Indeed, I found that including them does not improve the model performance. I also found that using **Rcuisine** (of the restaurants) does not improve the result, either, and it significantly increases the dimensionality of the data and the computation time given the large amount of categories it has.

Note that data preproceesing has to be done before performing the data analysis and visualizations in this section. The details of data preproceesing will be described in Section 3.1

2.3. Algorithms and Techniques

I used matrix factorization-based algorithms [6] to predict ratings. The fundamental concepts of these algorithms are that each item is characterized by a vector of features (x); each consumer

preference is described by a vector of weights (θ) with the same dimension as x; and the predicted rating of an item-consumer pair equals the inner product of the two vectors ($\theta^T x$).

The two main approaches to design a recommender system are content-based filtering and collaborative filtering. In the case of content-based filtering, the values of x are already determined based on known information, leaving θ to be optimized. For collaborative filtering, both the x and θ are the parameters to be optimized. I implemented both methods in this project. For the content-based approach, I removed irrelevant features and transformed the categorical features to numerical before modeling. I implemented the cost functions of these two methods and performed optimization using "minimize()" from the SciPy package. Mathematically, the cost function (without regularization) of the problem equals the sum of square errors for all the pairs of true and predicted values in the training set. The final predictions are generated by a model whose parameters minimize the cost function.

Before fitting the model to the training set, one important procedure is to perform "mean normalization" [7] on the ratings. For each restaurant, the mean rating is subtracted from all the original ratings (in the training set), so the mean of the normalized ratings is 0 for all the restaurants. A model is trained against these normalized ratings, and the mean ratings will be added back to the model predictions. This technique scales the predicted ratings to roughly the correct scale. What a model learns and predicts is therefore the deviation from the mean rating as a function of users, instead of the absolute values of ratings. The mean rating of each restaurant is essentially a base value of the predictions. If mean normalization is not performed, the model would perform poorly. Another reason to use this technique is that if a consumer never rated anything, the predicted rating of a restaurant from him/her will be the mean value.

Lastly, I also ran a singular-value decomposition (SVD) model [8] using the Surprise package [9] in comparison with the results of the two models I implemented. Since SVD is a collaborative filtering algorithm, the training process only takes the user ID's, item ID's, and ratings as inputs.

2.4. Benchmark

The benchmark to which I compared my solutions is a model that simply takes the mean rating as the prediction. In other words, the benchmark model does not consider the consumer preference or the restaurant features. It makes recommendations based on only the existing ratings. The resulting RMSE and FCP of my test set are 0.824 and 0.571 (76/133), respectively. Figure 10 shows a boxplot to compare the true and predicted ratings. The description of my train/test split procedure will be presented Section 3.2.

3. Methodology

3.1. Data Preprocessing

There are a total of 938 restaurants and 138 consumers in the dataset. However, only 138 of the restaurants are rated. Thus, I only worked on the data of these restaurants given that it is not wise to recommend a restaurant without ratings. I first cleaned and combined the data, and then created the following matrices: (1) an $n_r \times n_f$ matrix ($\mathbf{X}$) for the restaurant features, where n_r (= 130) is the number of restaurants and n_f is the number of their features, (2) an $n_u \times n_p$ matrix ($\mathbf{U}$) for the consumer features, where n_u (= 138) is the number of consumers and n_p is the number of their features, (3) an $n_r \times n_u$ matrix ($\mathbf{R}$) with its elements equal 0 or 1, indicating whether a rating exists for a restaurant-consumer pair, and (4) three $n_r \times n_u$ matrices (**Y_overall**, **Y_food**, and **Y_service**) for the overall, food, and service ratings.

For the collaborative filtering approach, $\mathbf{R}$ and the three $\mathbf{Y}$ matrices are all we need. For the content-based approach, a selection of restaurant features was done using the analysis described in Section 2.2 with $\mathbf{X}$ and $\mathbf{U}$. The selected features are **alcohol**, **parking_lot**, **price**, and **smoking_area**. I used pandas dummy encoding to transform **alcohol** and **price** to numerical variables, and then dropped one variable from each of them to avoid the "dummy variable trap" [10]. Because **parking_lot** and **smoking_area** have ordinal nature[2] for car owners and smokers, respectively, I directly transformed their categories to integers. The values I used, however, were simply decided by some trail and error based on the performance on the test set. The one to one relationship between the original categories and the chosen integers are as follows:

parking_lot : {none : 0, public : 1, valet parking : 2, yes : 2}

smoking_area : {not permitted : −1, none : 0, section : 1, permitted : 1, only at bar : 1}

3.2. Train/Test Split

I separated the existing ratings into training/test sets using a 70/30 split. Similar to the matrix $\mathbf{R}$, I created another two matrices, **R_train** and **R_test**, for the training and test sets, respectively. Note that what I split are the ratings, not the restaurants or consumers. Specifically, 30% of the ratings from each consumer are randomly selected and assigned to the test set. The minimum number of ratings from a consumer is 3, leading to a 2/1 split. Additionally, I ensured that every restaurant and every consumer receives/gives at least two ratings in the training set. This constraint makes sure that the characteristic of every restaurant and consumer is learned.

Ideally, it is better to split the data into training/validation/test sets, where the validation set

[2] Although **price** has an obvious order, it was shown in Figure 7 that consumers with different budget have different orders of preference.

is used to tune the model, and the test set is used to check whether the chosen model also performs well on unseen data. However, the number of existing ratings in this dataset is really small. If I used, for example, a 60/20/20 split, most of the users in either the validation or test set would have less than three true ratings, which is not very useful for evaluating the predicted restaurant ranking.

3.3. Implementation

A function computing RMSE was implemented by taking the square root of the sklearn "mean_squared_error" function. For FCP, I implemented from scratch. To compute the FCP of the predictions for a sample of ratings, I used a for loop to scan through all the consumers and selected the ones with at least two true ratings. A second-layer for loop was used to scan through all the pairs of true/predicted ratings from a given consumer. The numbers of ties (of the true ratings), CP's, and DP's were summed in the for loops, and the value of FCP was obtained using Equation 2.

The cost function of the problem was computed using a vectorized implementation. Assuming the weights of a consumer is a vector θ and the features of a restaurant is a vector x, when a bias term b is used, the predicted rating for this consumer-restaurant pair is $\theta^T x + b$. Another way to use the bias term is to include b in θ and add a 1 into x, such that $\theta_0 = b$ and $x_0 = 1$. Using this notation, the predicted rating is $\theta^T x$, which is the notation I used. The cost function J can be expressed in the following equation, where "CF" means collabarative filtering; $X = \mathbf{X}$ (an $n_r \times n_f$ matrix) is the stack of all the x^T of the restaurants; $\Theta = \mathbf{Theta}$ (an $n_u \times n_f$ matrix) is the stack of all the θ^T of the consumers; λ is the L2 regularization coefficient; Y (either $\mathbf{Y\_overall}$; $\mathbf{Y\_food}$, or $\mathbf{Y\_service}$) is the matrix of the true ratings; R equals $\mathbf{R\_train}$; the operator $\circ$ represents a element-wise multiplication; and $\| \ \|^2$ computes the sum of squares of all the matrix elements:

$$J = \frac{1}{2} \left\| X\Theta^T \circ R - Y \right\|^2 + \frac{\lambda}{2} \left\| \Theta\left[:, 1:\right] \right\|^2 + \left(\frac{\lambda}{2} \left\| X\left[:, 1:\right] \right\|^2 \right)_{\text{if CF}} \tag{3}$$

Remember that for the content-based algorithm, only the elements of Θ are free parameters.

In order to minimize J, the gradient of the function with respect to the free parameters are required (although SciPy's minimize() is able to compute the gradient numerically). The gradient can be computed by:

$$\begin{aligned} \frac{\partial J}{\partial X\left[j,k\right]} &= \left[(X\Theta^T \circ R - Y)\Theta + (\lambda X) \right]\left[j,k\right] \ \ (k \neq 0; \text{only for CF}) \\ \frac{\partial J}{\partial \Theta\left[j,k\right]} &= \left[(\Theta X^T \circ R^T - Y^T)X + (\lambda\Theta)_{\text{if } k\neq 0} \right]\left[j,k\right] \end{aligned} \tag{4}$$

Because the parameters of a function have to be a 1D vector for SciPy's minimize() to operate, the initial parameters have to an unrolled version of $\mathbf{Theta}$ and $\mathbf{X}$. After they were passed into

the cost function and gradient function, they were reshaped into matrices again for vectorized implementation. I used random numbers drawn from a normal distribution with the mean = 0 and $\sigma = 1$ to initialize the parameters.

3.4. Refinement

In the rest of this report, I will show the results of predicting **Y_overall** only. The procedures for another two types of ratings are the same. To consider all the three ratings for making recommendations, one can predict each type of ratings separately and then sum the predictions. The total predicted ratings, with a range of 0–6, can then be used to rank the restaurants. Another approach is to sum the three ratings first and then predict the total ratings directly. However, I found that the results of these two approaches are quite consistent, which is possibly due to the fact that the three types of ratings highly correlate with each other.

The final result of the content-based algorithm is actually similar to the initial one, but the computation time is reduced from several minutes to less than 10 seconds. This is because more features (such as **Rcuisine**, **other_sevices**, and **dress_code**...etc) were used at the beginning. The RMSE of the test set decreased a little from 0.702 to 0.701, and the FCP improved from 0.579 (77/133) to 0.602 (80/133). This illustrates that FCP is a better metric for a recommender system. Note that the numbers quoted here are the optimal values in each case that were tuned by changing the regularization coefficient.

These results, however, are not very different from the benchmark in terms of FCP. In fact, the bias term of **Theta** dominates the results, and all the other weights have very small values. In other words, the main effort made by this model is that it found the deviation from the mean rating (i.e., how much above or below the mean rating) each consumer rated on average. If only the bias term is optimized and all the other parameters are 0, the resulting RMSE would be smaller than the benchmark, but the FCP remains the same. Therefore, my result suggests that the restaurant features in the dataset have very minor effect on the ratings. Using collaborative filtering should be a better approach.

For the collaborative filtering model, the initial solution was produced without the bias term. Tuning the number of features (n_f) and the regularization (λ) showed that the best performance can be achieved with $n_f = 3$ and $\lambda = 2$. This also much reduced the computation time, as my original guess for n_f was 10, which corresponds to $10 \times 130 + 10 \times 138 = 2680$ parameters (**X** and **Theta**). The RMSE and FCP of the test set are 0.784 and 0.602 (80/133), respectively. Increasing the number of features generally decreases RMSE but makes FCP worse. When the bias term was used, $n_f = 3$ (excluding the bias term) and $\lambda = 0.5$ lead to the best result. The RMSE and FCP of the test set are 0.760 and 0.647 (86/133), respectively. Figures 11 and 12 compare the true and predicted ratings of my two final models using boxplots.

4. Results

4.1. Model Evaluation and Validation

As described in Section 3.4, I used the two evaluation metrics for model evaluation and valuation. However, my model selection was done by optimizing solely the FCP of the test set. Figure 13 illustrates the variation of RMSE/FCP as a function of the regularization coefficient λ for the content-based approach, where $\lambda = 16$ leads to the highest FCP. The same types of curves for the collaborative filtering approach are shown in Figure 14 with either the regularization or the number of features as the variables, where $n_f = 3$ and $\lambda = 0.5$ leads to the best result.

We can see large gaps between the training and test scores in Figures 13 and 14, an indication of overfitting. For the content-based method, the result is likely the best that can be achieved with the given restaurant data. Including more features in the model does not result in a better FCP. Increasing λ to even larger values does not improve the performance, either. I conclude that the content-based approach can be significantly improved only if more decisive features of the restaurants are obtained. The collaborative filtering method has a better result because the model learns the features itself. However, the degree of overfitting is even worse than the content-based approach. Increasing λ to even larger values can reduce the gap between the training and test scores, but does not improve the FCP of the test set. If there were more existing ratings, a larger regularization might actually give a better result.

4.2. Justification

The two of my final results are both slightly better than the benchmark result. Because of the mean normalization procedure, the benchmark prediction is actually the baseline of my solutions. The benchmark model is in fact a common way to produce item recommendations in many websites or apps, i.e., showing a consumer the top-n list in the order of average ratings. This is already the best we can do for a consumer who has never rated any items. My collaborative filtering model should be a sufficient solution for this dataset. If there were more existing ratings, a different set of hyperparameters (n_f and λ) would likely optimize the performance, and the degree of overfitting could be reduced.

For comparison, I ran the famous SVD algorithm using the Surprise package. The main difference between this model and my collaborative filtering model is that it does not perform mean normalization prior to the training process. Instead, it fits one bias term to account for the baseline of each item, and another bias term for the consumers. The former essentially works similar to the mean normalization procedure. By randomly sampling the hyperparameters in 1000 iterations, the best FCP of the test set I obtained with this model is 0.617 (82/133), and the corresponding RMSE is 0.636. This is similar to my results. I therefore conclude that my models are robust enough for the problem.

5. Conclusion

5.1. Free-Form Visualization

Unlike a typical supervised learning task, the robustness of predictions of a recommender system varies between different users depending on the number of ratings they have given. This is because each consumer corresponds to a set of weights in the model. In my test set, the number of true ratings per user ranges from 1 to 5, corresponding to $\sim$ 2–12 in the training set. Thus, I split the users into five groups and used the two evaluation metrics to inspect the model performance individually. As shown in Figure 15, although there is some randomness, the two models generally perform better when there are more existing ratings.

5.2. Reflection

The process of this project can be summarized in the following:

1. The data were loaded, inspected, and preprocessed to several matrices.
2. The dependence of ratings on restaurant features and user features was analyzed.
3. Feature selection was performed for the content-based approach.
4. The ratings were split into a training set and a test set.
5. A benchmark model was created for comparison.
6. The content-based and collaborative filtering models were implemented and tuned.
7. The Surprise package was used to run a SVD model on the data for comparison.
8. The results were presented and discussed.

There are some difficulties in this project. The first one is the small amount of existing ratings, which makes it hard to split the data into training/validation/test sets as well as to use MAP to evaluate the top-n list predicted by the model. Secondly, the most relevant features are not in the dataset. Because the main factors affecting the restaurant ratings (i.e., food and service) are hard to be quantified and recorded into structured data, the collaborative filtering method performs better than the content-based one in terms of restaurant ranking. One interesting fact I found is that FCP behaves more randomly than RMSE, which we can see in Figures 13, 14 and 15. This is because my algorithms minimize the sum of square errors, which is equivalent to minimizing RMSE. It would probably be useful to create a different cost function that computes the degree of consistency between the true and predicted restaurant ranking, as well as incorporates a regularization term.

5.3. Improvement

Some future work that can possibly improve the predictions for this restaurant-consumer dataset is summarized as follows:

- *Build a hybrid model.* Another type of recommender systems are hybrid models which combine both the content-based and collaborative filtering algorithms. Two of the many ways to implement a hybrid model are (1) combining the predictions from two separate models, and (2) incorporating the algorithms of both types into a single model. I have already attempted these approaches but have not obtained a better result yet.

- *Incorporate consumer features into the content-based model.* I have shown that the relevant restaurant features are the ones to which different consumers react differently. However, what I did was to let the content-based model learn the consumer preference (**Theta**) itself. Because there are already some useful consumer information, such as whether a person is a car owner or a smoker, one can attempt to use these known consumer features to tune the optimization. I have attempted to implement this concept, in which a user without a car does not have the weight corresponding to **parking_lot**, and a non smoker does not have the weight corresponding to **smoking_area**. However, this approach has not lead to a better result so far.

- *Use multiple regularization terms for the collaborative filtering model.* It is possible that using two regularization coefficients for **X** and **Theta** separately can result in better results. Adding another regularization coefficient for the bias term in **Theta** (the bias term is not regularized in my two models) also worth trying.

- *Develop a different cost function.* As discussed in Section 5.2, a cost function which computes the degree of consistency between the true and predicted restaurant ranking instead of the sum of square errors might be useful.

To test the robustness of my collaborative filtering algorithm, I will attempt to run it on other consumer-item datasets, such as MoveLens [11], although better computation resourcees or faster optimization algorithms will be required.

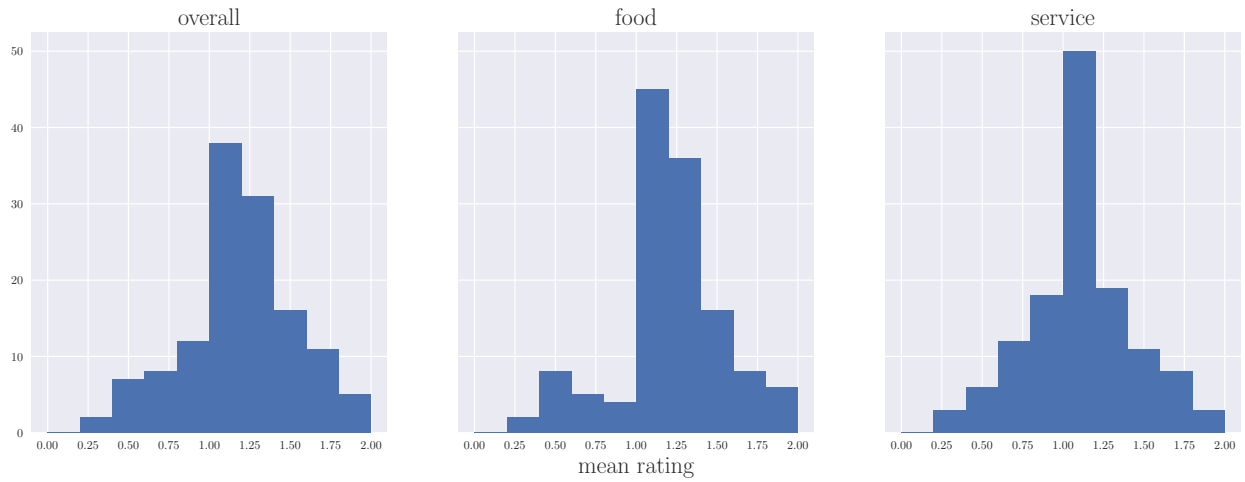

Fig. 1.— Distributions of the mean overall/food/service ratings of the 130 restaurants.

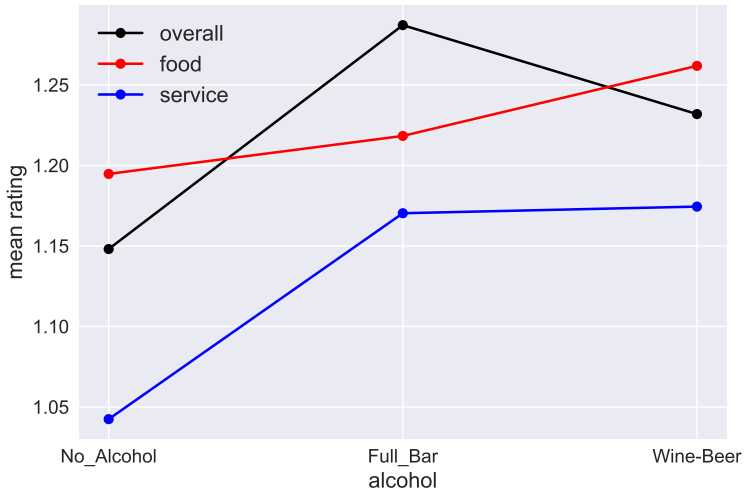

Fig. 2.— Dependence of mean ratings on **alcohol**. The food rating is least sensitive to this feature.

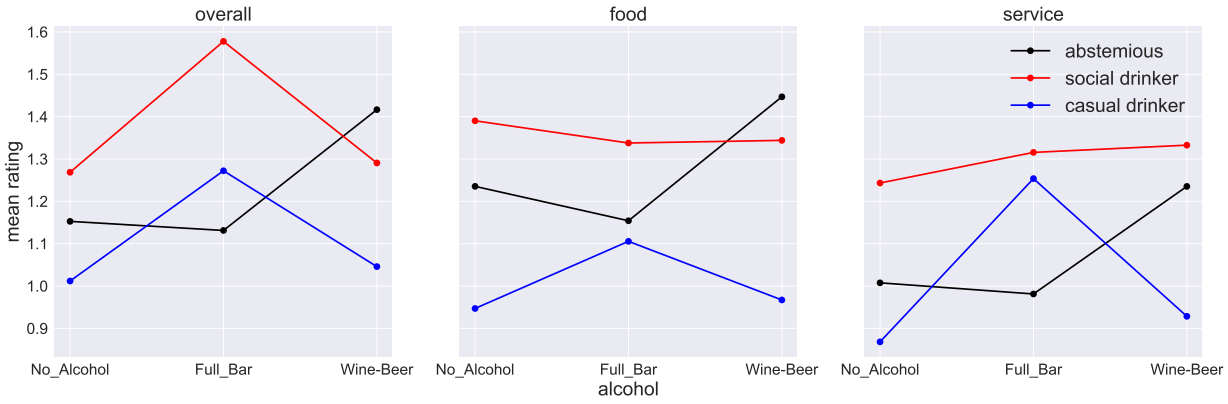

Fig. 3.— Dependence of mean ratings on **alcohol**, split into three type of drinkers, "abstemious", "casual drinker", and "social drinker". Observations: Abstemious drinkers prefer **Wine-Beer**; casual drinkers prefer **Full_Bar**; social drinkers prefer **Full_Bar** in terms of the overall rating as well, but this trend is not observed in the other two types of ratings.

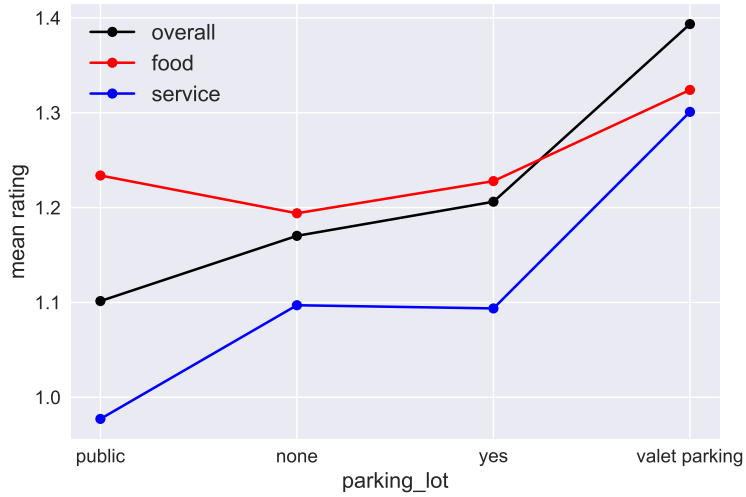

Fig. 4.— Dependence of mean ratings on **parking_lot**. The food rating is least sensitive to this feature.

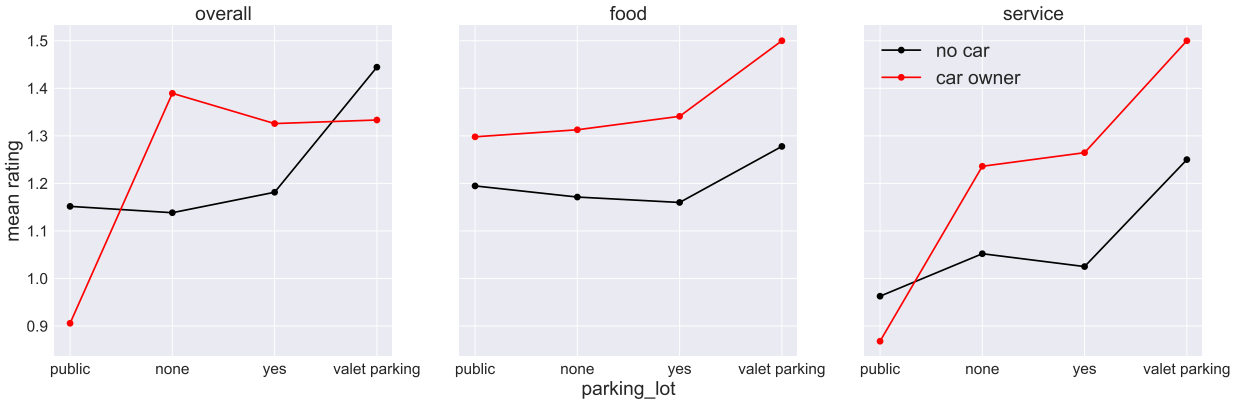

Fig. 5.— Dependence of mean ratings on **parking_lot**, split into car owners and non car owners. Observations: (1) Public parking is the least popular option and is worse than no parking, while valet parking is the most popular one. (2) Food rating changes the least among all three types of ratings, which is reasonable since it should not be affected by parking options at all. (3) The trend of car owners is only significantly different from non car owners in the overall ratings.

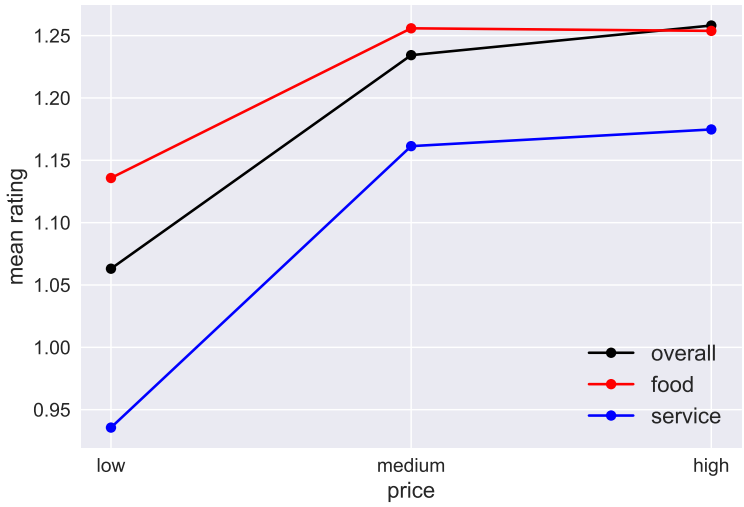

Fig. 6.— Dependence of mean ratings on **price**. Restaurants with medium or high price have higher ratings. The variation of food rating is the least.

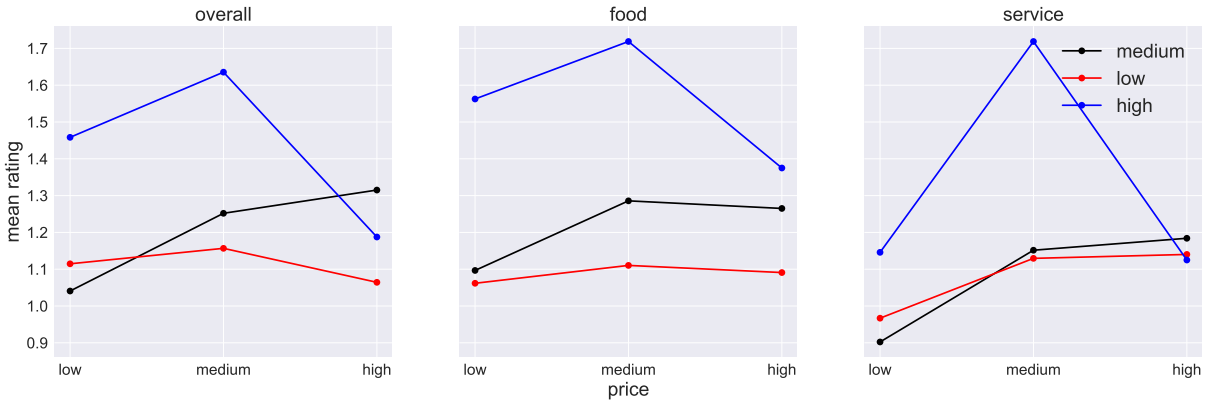

Fig. 7.— Dependence of mean ratings on **price**, split into consumers with low, medium, and high budget. Observations: The trend of consumers with high budget is significantly different from the others.

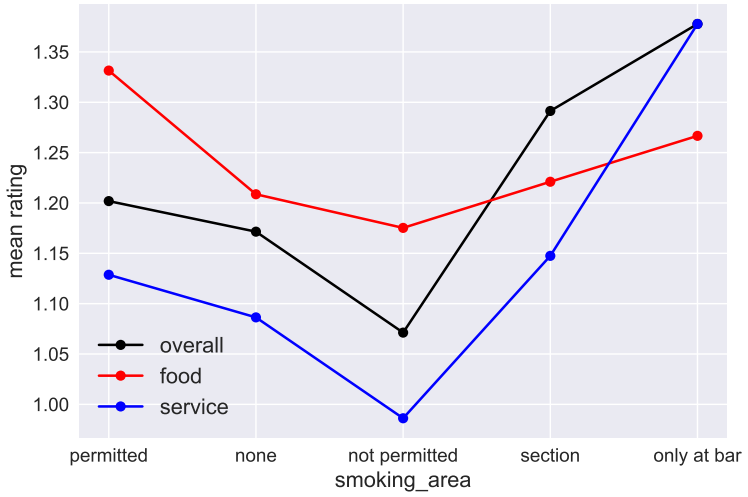

Fig. 8.— Dependence of mean ratings on **smoking_area**. The restaurants that do not permit smoking have the lowest ratings. The variation of food rating is the least.

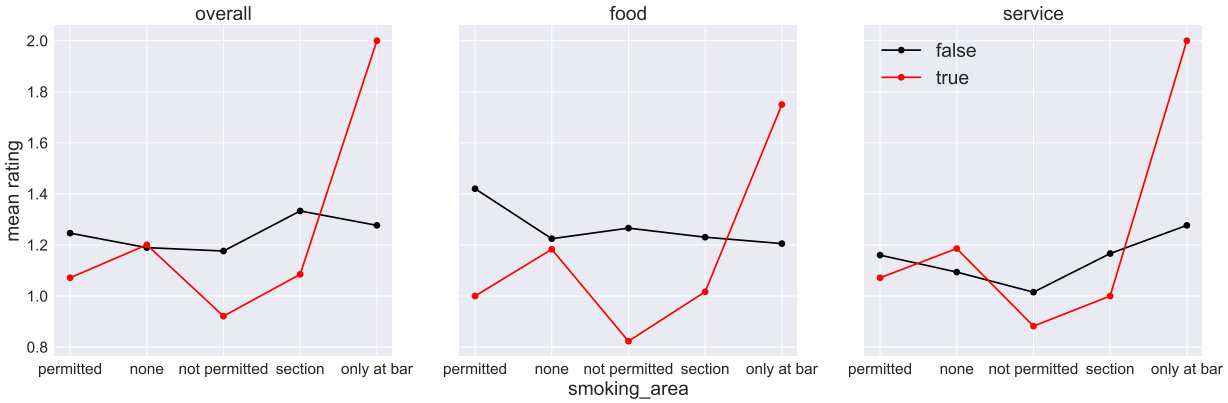

Fig. 9.— Dependence of mean ratings on **smoking_area**, split into smokers ("true") and non smokers ("false"). Observations: (1) There is nearly no dependence for non smokers. (2) For smokers, "only at bar" is very popular, while "not permitted" leads to very low ratings. (3) "None" and "not permitted" have very different mean ratings. This is probably because smokers can still find a place to smoke at those restaurants without a smoking area, but are completely forbidden at those "not permitted" ones.

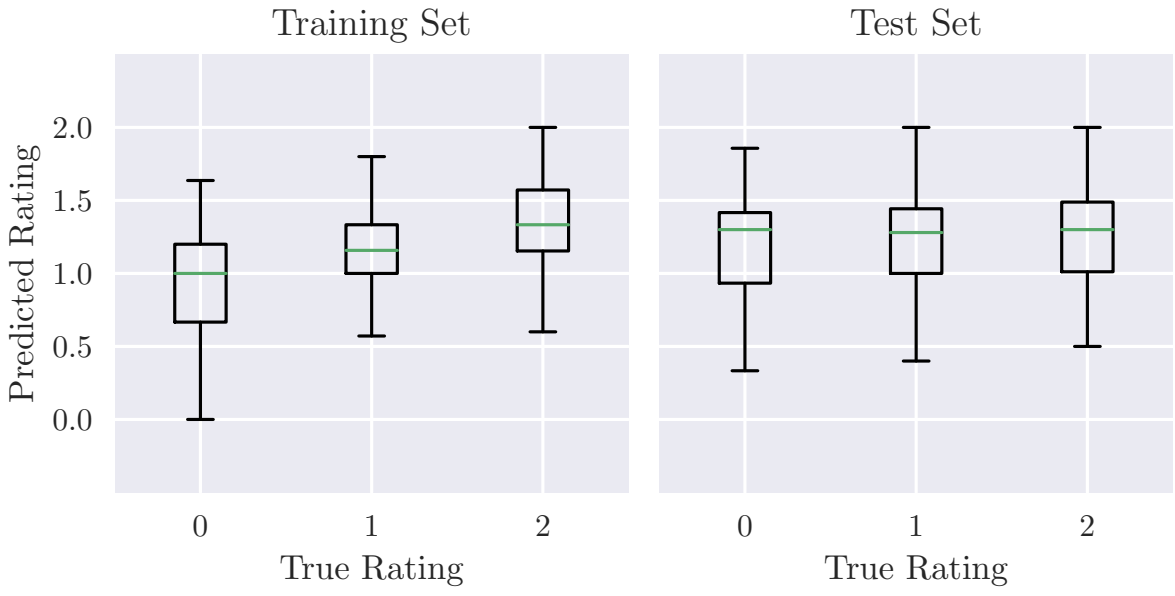

Fig. 10.— Comparison between the true ratings and the predicted ratings from the benchmark model. Boxplots are used to show the distribution of predicted values.

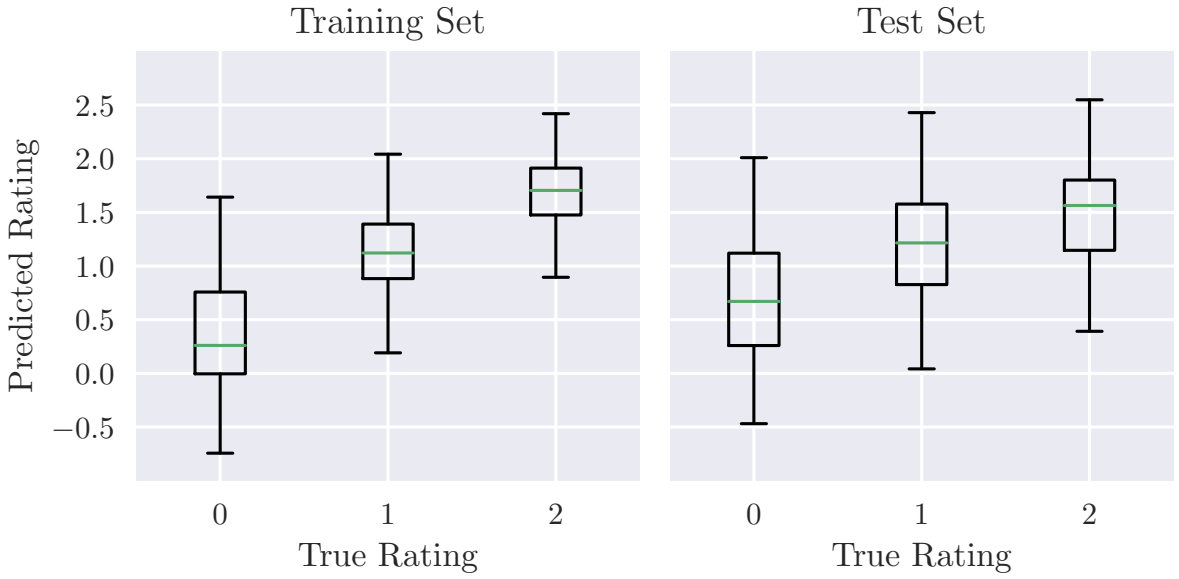

Fig. 11.— Comparison between the true ratings and the predicted ratings from the content-based model. Boxplots are used to show the distribution of predicted values.

Fig. 12.— Comparison between the true ratings and the predicted ratings from the collaborative filtering model. Boxplots are used to show the distribution of predicted values.

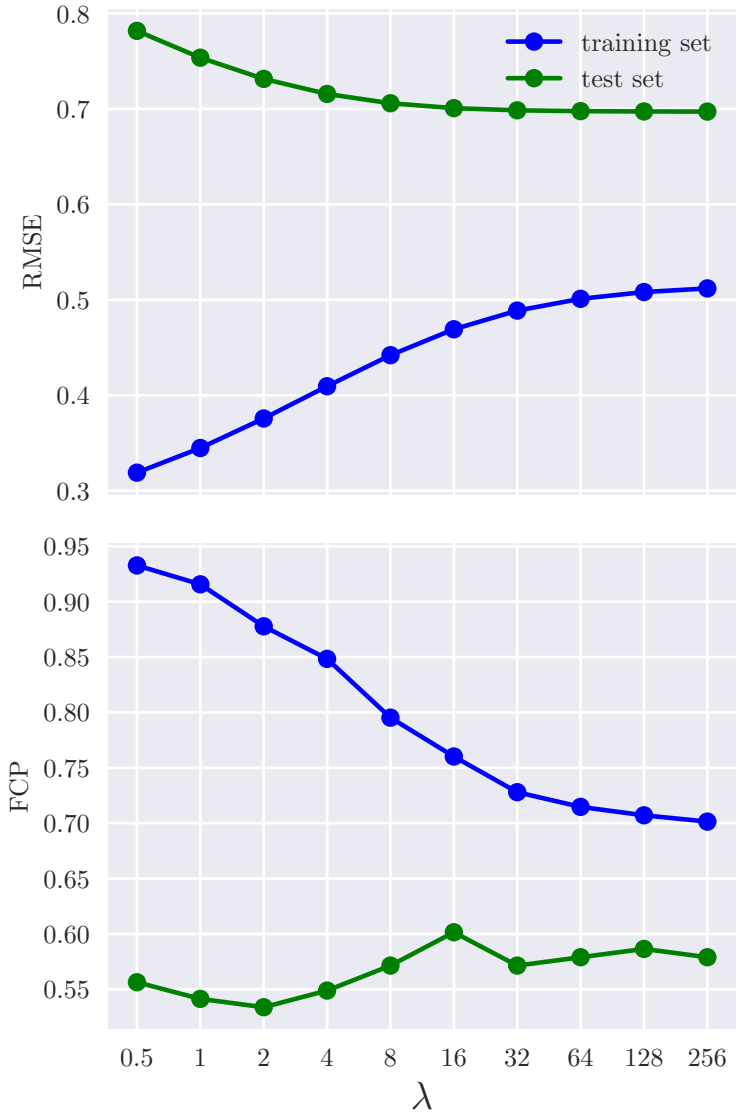

Fig. 13.— Variations of RMSE and FCP with respect to the regularization coefficient (λ) for the content-based model. The final solution was chosen at $\lambda = 16$, which leads to the highest FCP of the test set.

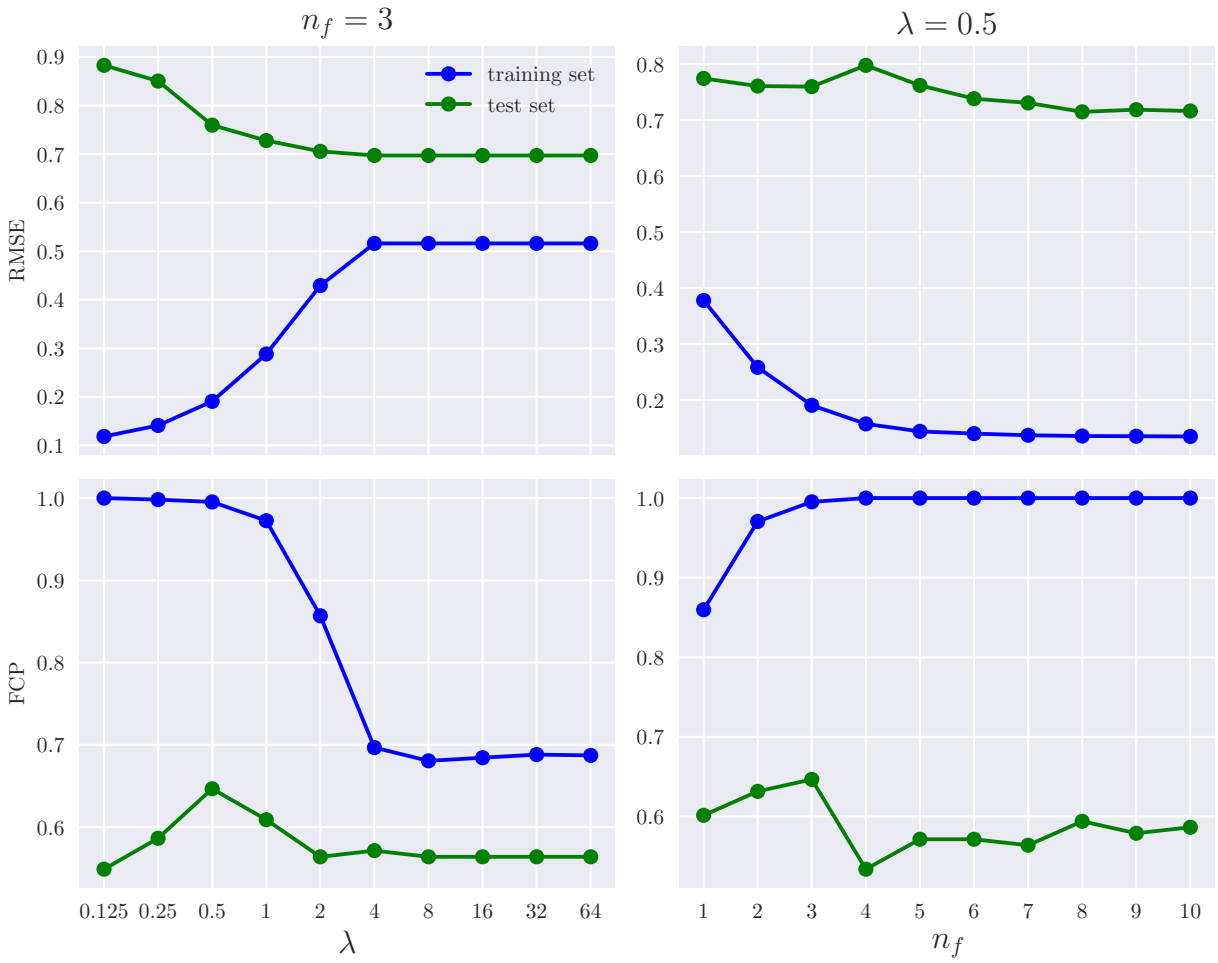

Fig. 14.— Variations of RMSE and FCP with respect to the regularization coefficient (λ) and the number of features used (n_f; excluding the bias term) for the collaborative filtering model. The final solution was chosen at $\lambda = 0.5$ and $n_f = 3$, which leads to the highest FCP of the test set. For illustration purposes, n_f is fixed at 3 in the left panels, and λ is fixed at 0.5 in the right panels.

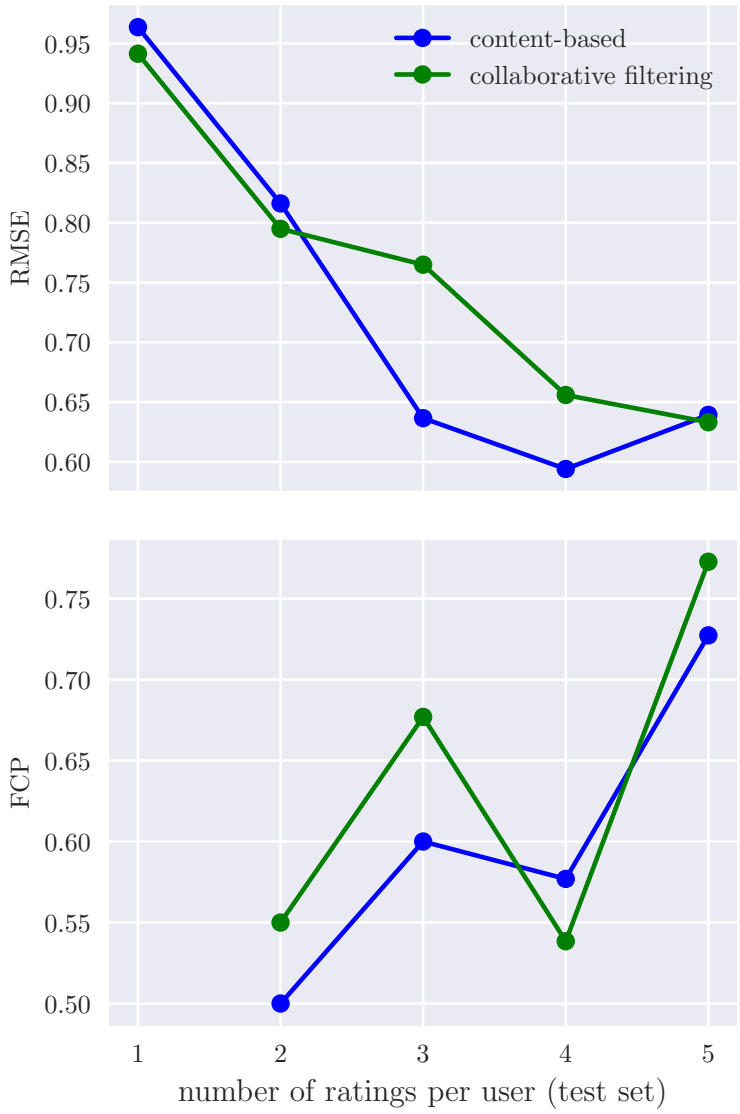

Fig. 15.— Variations of RMSE and FCP for the test set with respect to the number of ratings per user for the two models. Although there is some randomness, the two models generally perform better when there are more existing ratings. Note that for the consumers with only one rating, FCP cannot be computed.

References

[1] https://en.wikipedia.org/wiki/Recommender_system#cite_note-7

[2] https://archive.ics.uci.edu/ml/datasets/Restaurant+%26+consumer+data#

[3] http://ceur-ws.org/Vol-791/paper8.pdf

[4] http://www.ijcai.org/Proceedings/13/Papers/449.pdf

[5] https://en.wikipedia.org/wiki/Information_retrieval

[6] https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf

[7] https://www.coursera.org/learn/machine-learning

[8] http://sifter.org/simon/journal/20061211.html

[9] http://surpriselib.com/

[10] http://www.algosome.com/articles/dummy-variable-trap-regression.html

[11] https://grouplens.org/datasets/movielens/