Rating Prediction for Google Local Data

Long Jin A53056829, Xinchi Gu A53054946

In this project, we analyze the Google Local Dataset. First, we analyze the characteristics of the dataset and then focus on the rating prediction problem in recommender systems. Our goal is to leverage the rich information in this dataset to improve the rating prediction. First, to leverage the location information, we study the correlation between the rating of one business and its neighbours, which leads to the Location based Latent Factor Model (LLF). LLF would achieve similar performance as SVD++, in terms of RMSE and MAE in rating prediction. Then, to leverage the review text information, we treat the rating prediction problem as a multiclass classification problem and we find the review text is very useful to predict the ratings.

Dataset characteristics

We are using Google Local data set for this experiment that reveals the basic information of different business entities around the globe, and also the ratings from users for these business entities. These information are captured in three files: places.json, users.json and review.json.

- **places.json**: name, id, hours, phone, closed, address, gps
- **users.json**: userName, currentPlace, education, jobs, previousPlaces
- **reviews.json**: userName, rating, review, categories, gPlusPlaceId, texttime, gPlusUserId, utime

We visualize the whole dataset in Figure 1 and we can see its data spans all over the world. We also generate 3 subsets of the entire data set to conduct the experiment with in hope to see a variation of RMSE given the variation of density in our dataset. As is shown in the later part of the report, the denser the data set is, the more accurate the prediction will be.

Table 1: *Dataset Information*

Data	Users	Places	Reviews
Original	3,747,939	3,114,353	11,453,845
Dataset A	35,582	21,488	444,148
Dataset B	8,602	3,739	118,696
Dataset C	2,175	1,239	34,197

We generate dense data by getting rid of the sparse data, and we have generateed different levels of density based on the number of ratings that a user has given, and also the number of ratings that a business has received. We did multiple tests on the best density level, and have come to a conclusion that the below generations of density level will lead the the best experiment result.

- Dataset A: All users, places and reviews are trimmed down so that only users who have given greater or equal to 20 reviews and business that have received at least 20 reviews are selected.
- Dataset B: Same as above but with only users and places that have more than 50 reviews are selected.
- Dataset C: Same as above but with only places that fall in the "Restaurant" category. Shown in Figure 2

The original dataset as shown in Figure 1 spans across all the 5 continents and across tens of thousands of categories and users. This original data lead to a majority of sparse data that would hardly contribute to the accuracy of the model training.

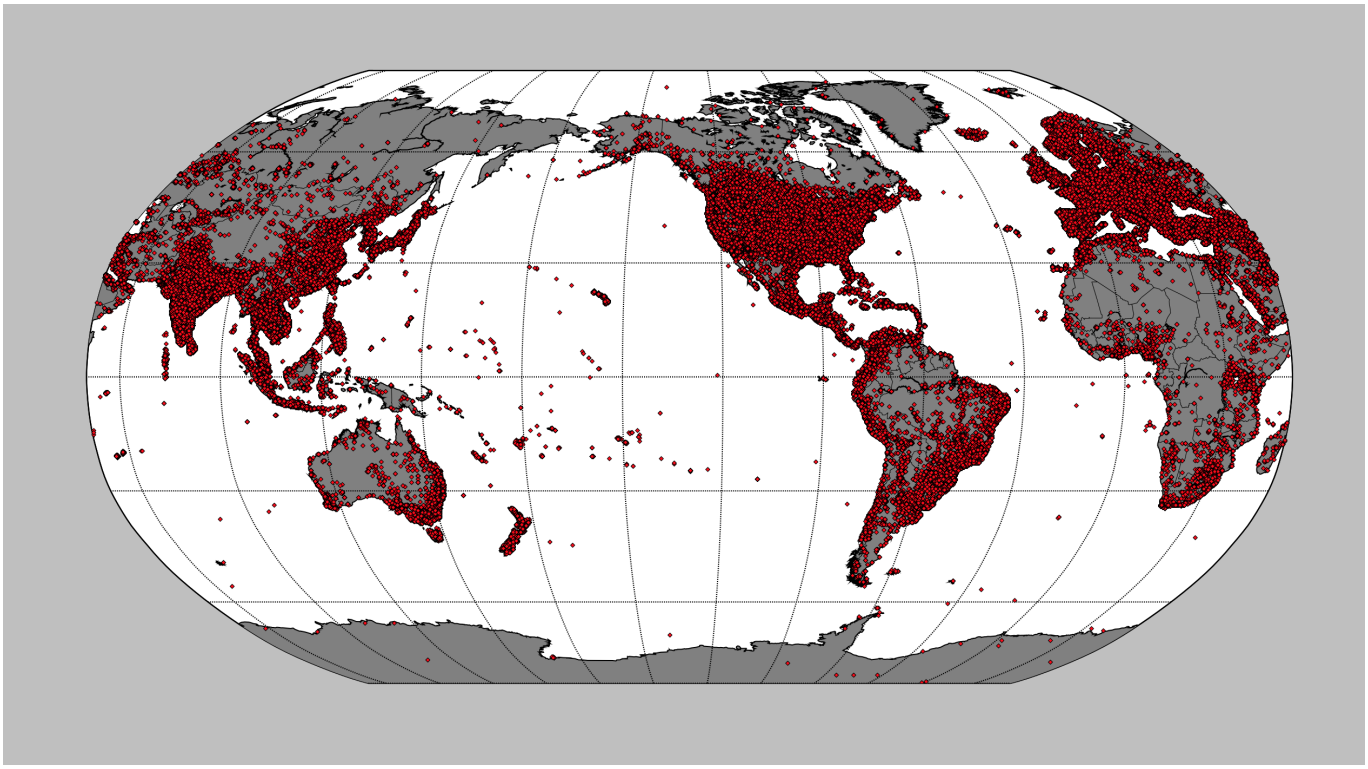


Figure 1: *Visualization of Original Dataset*

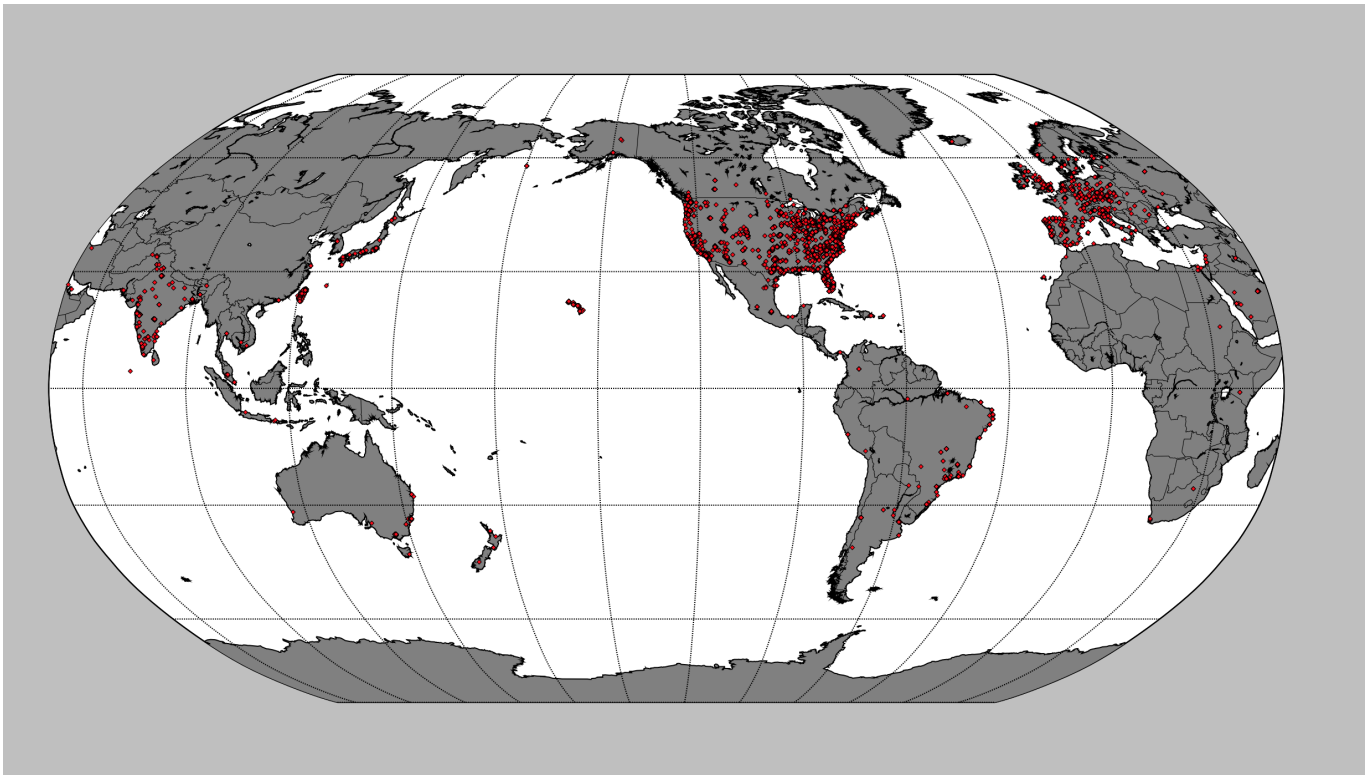
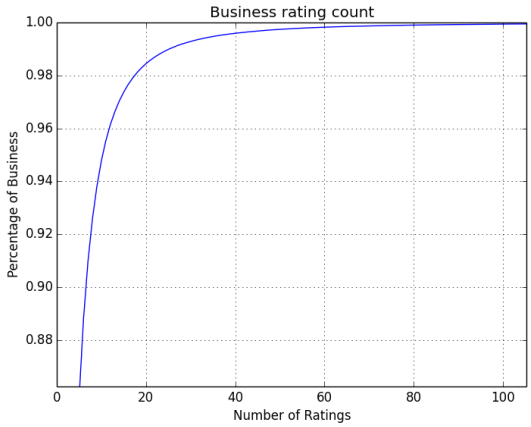


Figure 2: *Visualization of Dataset C*

Table 2: *Top 10 Categories in Google Local Dataset*

Category	Count
Restaurant	1,116,843
Hotel	543,819
European Restaurant	525,714
Asian Restaurant	473,580
American Restaurant	433,976
Italian Restaurant	423,744
Bar	364,460
Pizza Restaurant	333,270
Fast Food Restaurant	307,823
Cafe	257,379

After discarding some sparse data based on the methodology introduced above, we are left with the review data that are given by users that have given at least 50 reviews, of which the business have received at least 50 ratings as shown in Figure 2. As will be shown in the later part of the report, such dense data contributes greatly to the accuracy of the model training.



W

Figure 3: *Business Rating Percentile*

After parsing the data, we realized that on the original data set, a majority of the users didn't give more than 10 ratings, and a majority of business (merchants) didn't receive more than 10 ratings either. Figure 3 shows that more than 90% of the business receive less than 10 ratings, and as Figure 5 shows that more than 97% of the users give less than 10 ratings With the fear in mind that such sparse ratings might negatively affect the training accuracy, we discarded the data that is too sparse, and leave behind only those data that meet the below criterion: users that have given at least 50 ratings and business that have received at least 50 ratings. To make the experiment more robust we have also parsed data

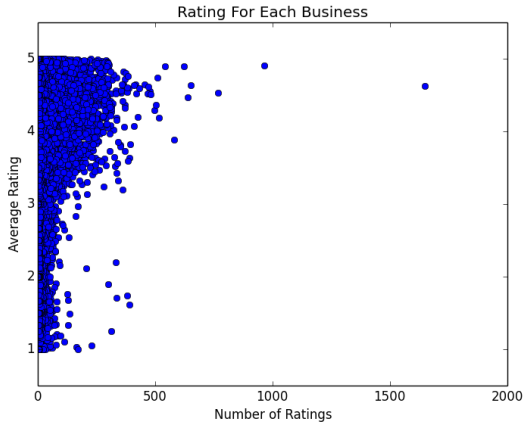


Figure 4: *Individual Business Rating*

for all business that have received more than 20 reviews and users that give at least 20 reviews across all business categories, and also the same dataset but only within the "Restaurant" category which contributes the most number of reviews. It can be clearly observed from Figure 2 that a majority of the business that received more than 50 reviews are located in either North America region or European region.

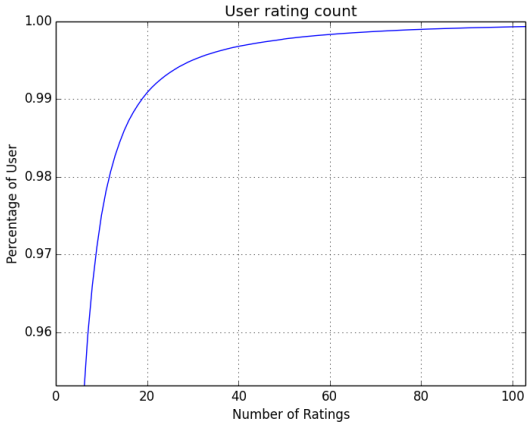


Figure 5: *User Rating Percentile*

Another conclusion that we got is that according to Figure 4, the business with more ratings tend to have higher ratings.

From Figure 6 we can see that most business have all the 5 closest business within a distance of $10\hat{1}(10)$ meters. This is an important assumption that has been verified because our model is based off of the assumption that each business that we are investigating has some nearby neighbors that are not too far apart.

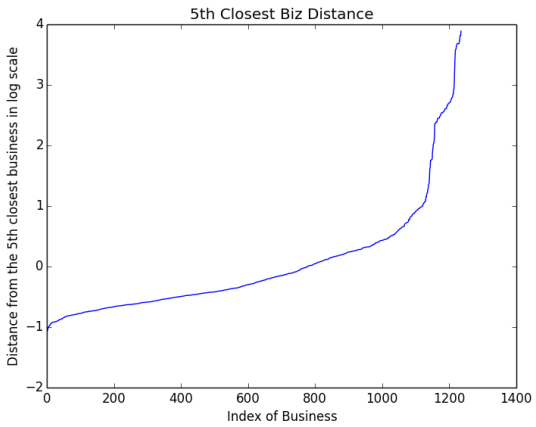


Figure 6: *Distance apart from the 5th closest biz*

The number of business in each category is shown in Figure 8. We reazlied that the number of business decreases sharply after the top few categories. So that is why we chose to conduct experiment on both the business across all categories, and across the top category - "Restaurant" to make the data more dense. As in the later part of the experiment we can see that the denser the data is, the smaller MSRE it will lead to.

In Figure 7, we have extracted out all the data based on the type of the restaurant. The graph shows the number of reviews for each individual restaurant type, and their respective average review. We can infer from the result that Japanese Restaurant has the best average rating out of all the restaurants (4.102) and that American restaurant has the most number of branches around the world(433976). It seems that the restaurants with higher price tends to have a higher rating, like Japanese restaurant and Seafood restaurant. However, we don't have the price information for each restaurant, so we could not validate our assumption.

Rating Prediction Problem

In the project, we are interested in building the rating prediction model for the Google Local dataset. For traditional recommender systems, like those in the Netflix competition, they would make the rating prediction only based on the ratings and few meta information about the users and the items are provided. However, in our Google Local dataset, each business, user and review have rich information. For example, each business has its location and each review has the review text. So we hope to leverage on such rich information to make our rating prediction to be more accurate.

Here, we use u to indicate user and i to indicate business/item, and r_{ui} to indicate the rating which is given to business/item i by user u.

For the baseline estimates, we consider the following three methods.

Global Average (GA): This method simply uses the global average of all the ratings as the predicted value. If we don't know information, especially for new users or new items, it is reasonable to just give it the global average value.

$$b_{ui} = \mu$$

Latent Factor Model (LF): This model is discussed in our class. And it is effective and widely used. It performs recommendation by projecting users and items into low-dimensional spaces.

$$\widehat{r_{ui}} = \mu + b_u + b_i + {p_u}^T q_i$$

Also, we add the regularization terms to the optimization problem as

$$min \sum_{(u,i)} (r_{ui} - \widehat{r_{ui}}))^2 + \lambda_1(||p_u||^2 + ||q_i||^2) + \lambda_2(b_u^2 + b_i^2)$$

Here, p_u and q_i are both vectors and represent the latent factors for user u and business i, respectively. Usually, we need to use stochastic gradient descent (SGD) method to train the model and the update rule is as follows and γ is the learning rate.

$$b_u \leftarrow b_u + \gamma(e_{ui} - \lambda_2 b_u)$$

$$b_i \leftarrow b_i + \gamma(e_{ui} - \lambda_2 b_i)$$

$$p_u \leftarrow p_u + \gamma(e_{ui} q_i - \lambda_1 p_u)$$

$$q_i \leftarrow q_i + \gamma(e_{ui} p_u - \lambda_1 q_i)$$

SVD++ model: SVD++ is proposed during Netflix Prize by Koren and won the Netflix Prize. SVD++ integrates the explicit and implicit user feedback. The explicit user feedback includes the user ratings in the training dataset and the implicit user feedback includes whether the user rates a item. For instance, we could know the movies which are rated by the user even they are in the test data.

So, in SVD++, the user latent factor consists of two parts, one part is the individual user factors and the other part is expressed by the itmes that the user rated. Then,

$$\widehat{r_{ui}} = \mu + b_u + b_i + q_i^T (p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j)$$

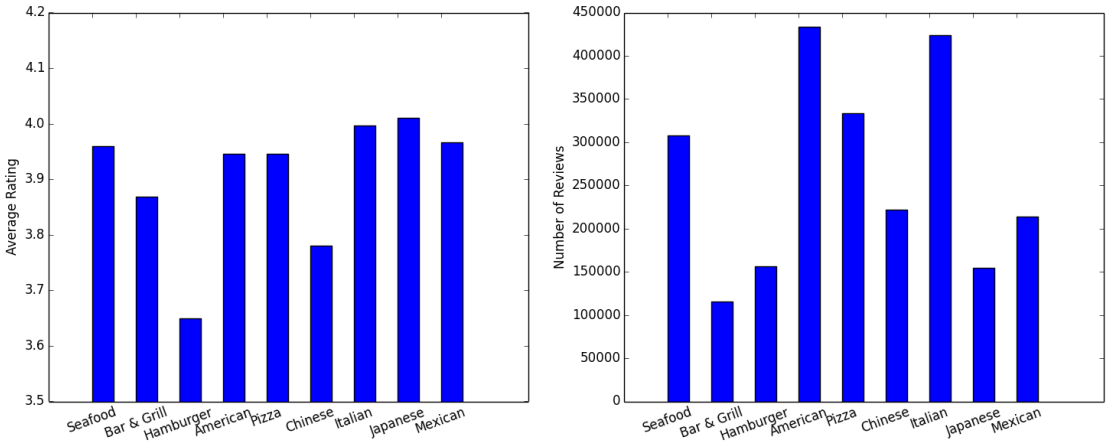


Figure 7: *Average Rating and # of Rating for Individual Restaurant Type*

. Here, p_u indicates the individual user factors for user u and y_j indicates the impact to the user factors when rating item j, and $N(u)$ indicates all the items which are rated by u in both the training data and testing data. We plug the y_j into the regularization and the optimization problem becomes

$$min \sum_{(u,i)} (r_{ui} - \widehat{r_{ui}}))^2 + \lambda_1(||p_u||^2 + ||q_i||^2) + \lambda_2(b_u^2 + b_i^2)$$

$$+\lambda_3 \sum_{j \in N(u)} ||y_j||^2$$

We also need to use SGD to train the model and the update rule is as follows.

$$b_u \leftarrow b_u + \gamma(e_{ui} - \lambda_2 b_u)$$

$$b_i \leftarrow b_i + \gamma(e_{ui} - \lambda_2 b_i)$$

$$p_u \leftarrow p_u + \gamma(e_{ui} q_i - \lambda_1 p_u)$$

$$q_i \leftarrow q_i + \gamma(e_{ui}(p_u + |N_u|^{-\frac{1}{2}} \sum_{j \in N(u)} y_i) - \lambda_1 q_i)$$

$$\forall j \in |N_u|, y_j \leftarrow y_j + \gamma(e_{ui}|N_u|^{-\frac{1}{2}} q_i - \lambda_3 y_j)$$

To compare and evaluate different models, we will calculate the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) on the test data to compare different methods. Given the test data set as T, then their definitions are as follows.

$$RMSE = \sqrt{\frac{1}{|T|} \sum (r_{ui} - \widehat{r_{ui}})^2}$$

$$MAE = \frac{1}{|T|} \sum |r_{ui} - \widehat{r_{ui}}|$$

Related Work

Rating prediction is one of the most important problems in the recommender systems. It focuses on modeling a two-dimensional relationship between user and items. The Netflix Prize has a big impact on this research topic and brought many interesting solutions. Collaborative filtering model[1] performs recommendation in terms of user/user and item/item similarity. Latent factor models and SVD-based models are also widely used and very effective, which performs recommendation by projecting users and items into low-dimensional spaces. Koren, who is the winner of Netflix, proposes SVD++ [2] which considers the implicit user feedback, i.e. the rating actions in both the training data and test data.
As the online social networks, like Yelp and Facebook, become popular, more and more review information is provided to build better recommender systems. Also, Yelp launch its own data challenge competition. There are many interesting work. McAuley etc.[3] uses the hidden topics learnt from the review text to regularize the latent factor model, which would be helpful to understand the latent factors and improve the rating prediction. Hu etc.[4] considers the geographical neighbourhood influence and observe the weak positive correlation between the ratings and its neighbours, which are also observed in our Google Local Dataset. Ma etc.[5] leverages on the social network information to improve the recommender systems and the key ideas are that similar users should have similar latent factors, which could be used in the regularization term.

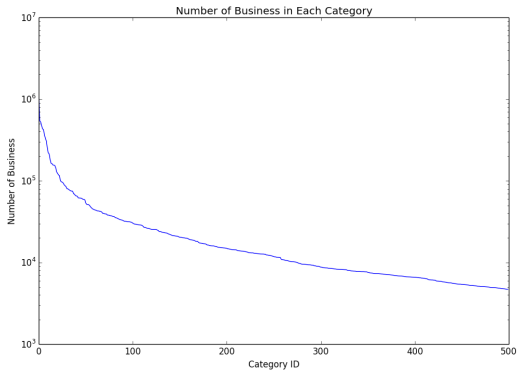


Figure 8: *# of Business in Each Category*

Identify Useful Features

Firstly, we consider how to leverage the business location information. The reason why we select the location is that location is one of the most important factors which would affect the success of the business. So people usually are very careful to select a proper location for their business. Also, location information is special in the Google Local Dataset in comparison to the Netflix Dataset, since movie doesn't have the "location" attribute. So, we are interested in how to use the location information to enhance the rating prediction.
Then, we study the correlation between the ratings of the business and its neighbours and plot it in Figure9. In each graph of Figure 9, X-axis represents the rating of each individual business, and Y-axis represents the average rating of the 3 (or 5) closest business. From the graph it is observed that each business rating is correlated to the nearby business's average rating, especially when we only limit to the same category ('Restaurant'). So we regard there are some positive impact by the neighbours, which is also observed in the Yelp Dataset by [4]. Usually, business would like to gather together and form a community and then to attract more users to visit the community. Also, better location would attract more business and better business to join, since location is one of the key factors which would determine the success of the business. So, we would like to integrate the neighbour influence into the latent factor model. We are also interested in how the review text would explain the review ratings. Our questions is that: could we predict the rating accurately only based on the review text? Review text contains the sentiment of the user and would express the opinions of the users. One interesting thing is that we could form this problem as a multiclass classification problem, since the ratings could only be the integers among 1 to 5. For the text mining and analysis, we plan to use the bag-of-words model. Based on the corpus of the review texts, we select the most frequent words to construct the feature. We think that there should be some useful words in this constructed feature, which would represent the opinion of the users.

Methdology

As discussed above, we are interested in two models for the rating prediction problem.
In the first model, we hope to leverage on the location information of the business. We call it Location based Latent Factor Model (LLF), in which we consider the influence caused by the neighbours. In this model, we decompose item latent factors into two components, one is business internal factors and the other is the influence caused by the neighbour business. It means that business would have some effect on its neighbour business. SVD++ focuses on modeling the user factors through item, and we focus on modeling the items by its implicit factor and explicit factor (affecting neighbours). So,

$$\widehat{r_{ui}} = \mu + b_u + b_i + (q_i + \frac{1}{|M(i)|}\sum_{n \in M(i)} |s_n|)^T p_u$$

. Here, $M(i)$ indicates the neighbours items of item i and s_n indicates the impact of item n to its neighbours. Then the optimization problems becomes

$$min \sum_{(u,i)} (r_{ui} - \widehat{r_{ui}}))^2 + \lambda_1(||p_u||^2 + ||q_i||^2) + \lambda_2(b_u^2 + b_i^2)$$

$$+\lambda_3 \textstyle\sum_{j \in M(i)} ||s_n||^2$$

We need to use SGD to train this model and the update rule is as follows.

$$b_u \leftarrow b_u + \gamma(e_{ui} - \lambda_2 b_u)$$

$$b_i \leftarrow b_i + \gamma(e_{ui} - \lambda_2 b_i)$$

$$p_u \leftarrow p_u + \gamma(e_{ui}(q_i + \frac{1}{|M(i)|}\sum |s_n|) - \lambda_1 p_u)$$

$$q_i \leftarrow q_i + \gamma(e_{ui} P_u - \lambda_1 q_i)$$

$$\forall n \in N_i, s_n \leftarrow s_n + \gamma(e_{ui} p_u \frac{1}{|M(i)|} - \lambda_3 s_n)$$

LLF model doesn't need to integrate the users' implicit feedback, so it doesn't need to look into the test data, and would focus on the training data only.

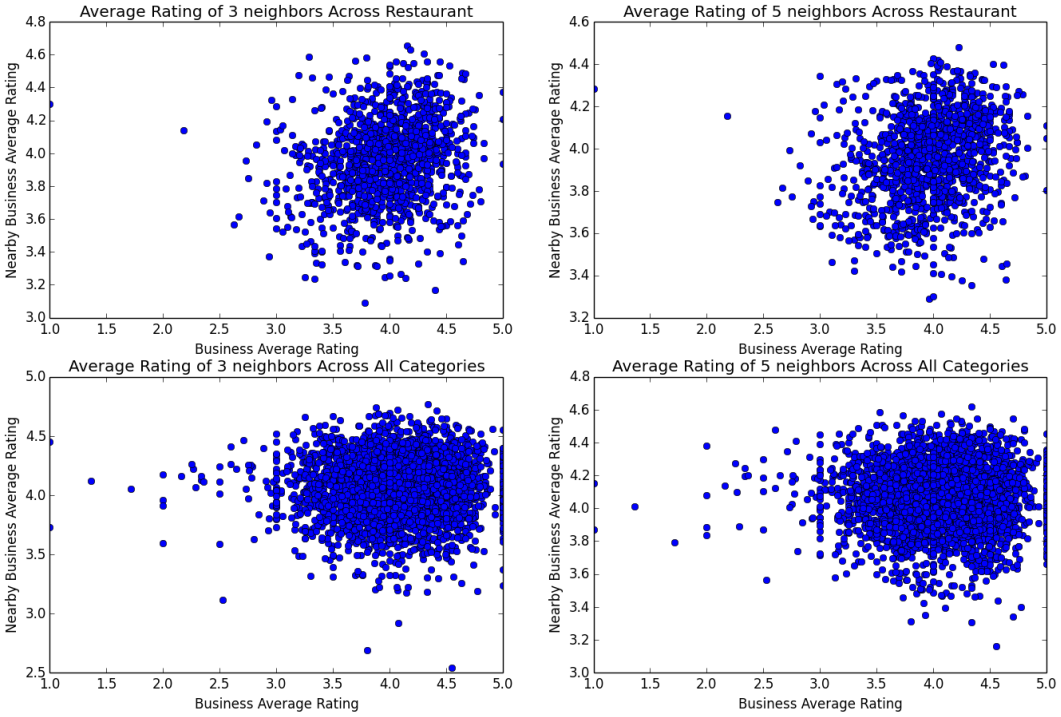


Figure 9: *Business Rating and its Nearby Business Average Rating*

In the second model, we only consider the review text and treat the rating prediction problem as a multiclass classification problem, since the rating could only be one of the integer among 1 to 5. The Google Local Dataset spans widely all over the world, so we only focus on the English review analysis. Here, if one review contains more than half of the words are belong to English, we treat the review as an English review. We analysis all the English reviews and use the bag-of-words model to choose the feature. We select the most frequent words as the bag of features. Then, each review could be represented by a multi-dimensional vector. We could study how the number of words in the bag would affect the performance of this modl.

We treat the ratings in the training data as the labels and train a classifier. We compare multinomial logistic regression and random forest model. When predicting the ratings, we require the review text, so this method is not feasible in real recommender systems. If we want to recommend some new items to users, usually we don't have their review texts. However, this model is still interesting since we can learn which words would fall into the bag of features and how effective it is to predict the ratings from the text.

Evaluation

We evaluate different models on the rating prediction problem. At the beginning, we are using the whole Google Local dataset, but we find the advanced methods didn't improve much over the global average method. We think that it is caused by the sparsity of the data. In Figure 3 and Figure 4, we can see that almost 98% of the users have fewer than 20 reviews and almost 99% of the business have fewer than 20 reviews. So, it motivates us to come up with Dataset A, B and C, which are more dense. For example, all the users and the business have more than 20 reviews in Dataset A, and all the users and the business have more than 50 reviews in Dataset B. The different between Dataset B and Dataset C is that the business in Dataset C are all in the restaurant category and we hope to see whether this will affect our model.

We previously split the whole dataset into training set and test set based on the natural order of the review, meaning that we would cut the first 80% away and make them the training set and have the rest 20% as the test set. However this methodology does not guarenttee randomness as the reviews could very likely come in chronologically, and those early users might have their review clustered at the first

Table 3: *Comparison of RMSE*

RMSE	GA	LF	SVD++	LLF
Original	1.1989	1.1066	1.1067	-
Dataset A	0.8895	0.8258	0.7805	0.7806
Dataset B	0.8559	0.7814	0.7549	0.7548
Dataset C	0.8237	0.7639	0.7352	0.7348

Table 4: *Comparison of MAE*

MAE	GA	LF	SVD++	LLF
Original	0.9551		0.8747	-
Dataset A	0.6625	0.6222	0.6028	0.6031
Dataset B	0.6439	0.5995	0.5858	0.5860
Dataset C	0.5808	0.5908	0.5706	0.5704

half of the whole review data set. Splitting the whole data set this way could lead to an unpredictable test set as a majority of users are woudl fall into either the training set or the test set, and not both, which would hamper the MSRE of the testing because the model knows nothing about a user if he doesn't appear in the training set. Instead of adopting such a methodology, we randomly select training dataset and test dataset to guarenttee a fair split.

For the Latent factor model, we set γ as 0.05, λ_1 as 0.3 and λ_2 as 0.15. For SVD++ and LLF, we set γ as 0.05, λ_1 as 0.5, λ_2 as 0.15 and λ_3 as 0.5.

Firstly, we compare the RMSE and MAE of different models and list the results in following tables. Here, we use GA to represent Global Average method, LF to represent Latent Factor Model and LLF to represent Location based Latent Factor Model.

In Table 3, we compare the RMSE of different models and we use the nearest 5 neighbours in the LLF model. We can see that LLF and SVD++ both outperforms GA and LLF. It is interesting that LLF and SVD++ achieve similar results, it may be caused that they are actually learning similar latent factors though by different ways. The relationship between these two methods need to be investigate more.

In Table 4, we compare the MAE of different models and the results are similar to RMSE. Also, we can observe that on the original dataset, even SVD++ would not improve a lot over the global average method. We think it is caused by the sparsity of the original dataset and it also satisfies our analysis in Section 1. Considering the whole dataset is too big, we don't compute the distance information on it, so we omit the result for LLF on the original dataset.

We also try to combine the LLF and SVD++, which means to use the location information to enhance the performance of SVD++. But their performance are almost similar, so we don't list the results for the combined model here.

Then, we compare the RMSE and MAE with different number of neighbours of the LLF model in Table 5-7. We can see that the number of neighbours would not affect the results too much on all the dataset.

In the second model, we use two different classifiers

Table 5: *LLF on Dataset A*

Neighbours #	1	3	5
RMSE	0.7805	0.7807	0.7806
MAE	0.6027	0.6037	0.6031

to fit the data. One is the Multinomial Logistic Regression (MLR) and the other is Random Forest (RF). First, we select 5000 most frequent words to construct the bag of features. In Table 8, we list the results when we remove all the stop words and in Table 9, we list the results when we keep the English stopping words. Since 'no' or 'not' is also considered as a stop word in English, we would like to see whether it would have some effect on the prediction accuracy. From the results, we can see Multinomial Logistic Regression performs better and the stop words would not have a big effect on the results.

We can see the logistic regression model would achieve the RMSE as 0.6308, which is even lower than the SVD++ results shown above. It is because the second model would predict the rating directly on the review texts and the review texts are the indicator of the users' opinion. However, when we would like to recommend some items to users, we usually would not have the opinions or texts from the user to the item, so this model would not be a feasible way for the rating prediction.

This method turns out to be effective to predict the ratings and we are interested in which words are used in the constructed feature. Then, we adjust the number of words in the bag to see its impact on the performance. In Table 10, we use multinomial logistic regression and adjust the number of words. And we could see set the number of words of 500 would achieve a good balance between accuracy and computation cost.

Table 6: *LLF on Dataset B*

Neighbours #	1	3	5
RMSE	0.7555	0.7549	0.7548
MAE	0.5866	0.5857	0.5861

Table 7: *LLF on Dataset C*

Neighbours #	1	3	5	10
RMSE	0.7350	0.7332	0.7348	0.7350
MAE	0.5701	0.5694	0.5704	0.5703

Table 8: *Results when removing stop words*

	Accuracy	RMSE	MAE
Logistic Regression	0.6324	0.6430	0.2151
Random Forest	0.5894	0.7214	0.2537

Here, we present the most popular 500 words (excluding the stop words) in all the English review text on page 10 and we could see many words with strong sentiment, like 'great', 'good', 'recommend', 'amazing', 'never', etc. So it is reasonable to get a good rating prediction based on these features.

Conclusion

We find that the rich information about the business and the users are very useful to make the rating prediction to be more accurate. To leverage the location information, we adopt the Location based Latent Factor Model (LLF), which would achieve similar performance as SVD++ and LLF would only use the information in the training data and no need to access the implicit feedback from the testing data. To leverage the review text, we use a multiclass classifier to make the rating prediciton and achieve very low RMSE and MAE.

Challenges

One challenge that we have faced was during the data parsing phase. The Google Local database is huge (multi-Gigabytes size for each file) and the memory no our computers are limited, and we have encountered "out of memory" cases that killed the process before the parsing has completed. Figure 10 One way to get around this issue is to parse the data in such a way that trades efficiency for memory usage: instead of reading in the whole file and store it in the memory by calling json.loads(file) command, we read the data line by line and dump the data in a secondary file that can be read in later on so that constant memory is used.

Table 9: *Results when keeping stop words*

	Accuracy	RMSE	MAE
Logistic Regression	0.6370	0.6308	0.2099
Random Forest	0.6040	0.7759	0.2664

Table 10: *Adjust the Number of Words in the Bag*

Number of Words	Accuracy	RMSE	MAE
100	0.5762	0.8875	0.3172
300	0.6055	0.7758	0.2661
500	0.6205	0.7213	0.2428
5000	0.6370	0.6308	0.2099

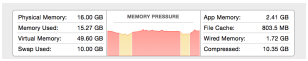


Figure 10: *Memory Usage During Data Parsing*

References

[1] Xiaoyuan Su, Taghi M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. Advances in Artificial Intelligence archive, 2009.

[2] Yehuda Koren. Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model. In ACM KDD, 2008.

[3] Julian McAuley, Jure Leskovec. Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text. In ACM RecSys, 2013.

[4] Longke Hu, Aixin Sun, Yong Liu. Your Neighbors Affect Your Ratings: On Geographical Neighborhood Influence to Rating Prediction. In ACM SIGIR, 2014.

[5] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, Irwin King. Recommender Systems with Social Regularization. In WSDM, 2011.

great, good, service, food, place, time, would, get, one, like, go, staff, best, back, car, always, nice, really, friendly, recommend, experience, even, us, never, work, also, well, people, new, went, got, ve, love, first, could, going, excellent, day, ever, make, price, made, customer, years, much, know, told, better, take, way, took, every, said, little, didn, amazing, store, came, come, restaurant, helpful, highly, right, care, family, want, called, around, home, everything, say, definitely, clean, room, two, done, looking, bad, find, many, order, see, need, job, times, area, business, dr, sure, awesome, prices, try, quality, feel, professional, found, pizza, another, still, location, lot, since, shop, help, asked, call, give, money, next, happy, last, anyone, minutes, long, wait, company, wanted, night, re, used, away, needed, old, look, hotel, manager, worth, wonderful, small, visit, everyone, eat, office, phone, year, use, atmosphere, friends, stay, though, think, guys, thank, pretty, problem, bar, things, thanks, ordered, nothing, something, free, gave, extremely, left, else, check, getting, days, put, delicious, able, owner, bit, big, different, enough, deal, chicken, far, rude, anything, menu, hours, week, several, top, without, needs, buy, person, vehicle, working, hour, pay, fantastic, ll, fast, selection, house, thing, wife, fresh, town, front, tried, coffee, ask, high, lunch, fun, guy, drive, let, tell, kids, easy, wasn, worst, places, however, hard, kind, sales, later, bought, keep, someone, perfect, quick, actually, school, months, super, knowledgeable, breakfast, customers, felt, full, local, whole, hair, absolutely, overall, comfortable, part, worked, recommended, less, beautiful, dealership, horrible, helped, couldn, almost, hot, ago, beer, life, quite, thought, questions, coming, couple, door, wrong, table, reasonable, live, received, water, meal, ok, services, purchase, reviews, drinks, favorite, waiting, dinner, must, loved, open, looked, team, busy, process, end, expensive, large, extra, real, friend, appointment, making, dog, rooms, walk, especially, half, special, second, poor, decided, side, close, least, return, makes, trying, buying, recently, walked, finally, cost, three, husband, repair, enjoy, today, stop, charge, terrible, weeks, employees, change, leave, within, review, past, won, brought, purchased, line, run, park, seen, fix, may, started, taste, month, parking, paid, city, outside, cars, taking, probably, entire, pick, taken, decent, pleasant, disappointed, honest, items, inside, problems, fixed, huge, cheap, drink, impressed, name, issues, cheese, bring, quickly, stayed, usually, fair, sushi, priced, wouldn, value, burger, arrived, offer, sauce, cut, stuff, set, enjoyed, treated, course, patient, fish, plus, yet, options, fine, doesn, already, cold, ready, music, management, morning, knew, point, issue, kept, show, given, available, delivery, spot, expect, instead, completely, tasty, son, waitress, trip, gone, pleased, please, lots, moved, bill, man, cool, dentist, truck, pain, restaurants, future, salad, oil, either, online, late, looks, doctor, believe, due, seemed, start, fact, twice, others, short, number, using, slow, group, street, truly, showed, daughter, saw, soon, wish, pool, often, decor, offered, eating, although, lady, children, beyond, meat, talk, maybe, anywhere, seems, waited, exactly, steak, reason, insurance, explained, seem, comes, wine, charged, courteous, card, class, fries, served, might, center, wedding, etc, choice, understand, dental, lovely, credit, dont, world, desk, move, ended, auto, party, met, bike, spent, outstanding, sandwich, provided, facility, isn, floor, style, sell