# Data Mining Techniques for Recommending Stores to Shopping Malls

John Wu
Department of Mathematics
New York University
Courant Institute
Email: jsw452@nyu.edu

Ying Zhang
Department of Computer Science
New York University
Courant Institute
Email: zy674@nyu.edu

Purnima Padmanabhan
Department of Computer Science
New York University
Courant Institute
Email: pp1492@nyu.edu

***Abstract*—In this paper, we try to recommend stores for malls. We do this by using using various traditional recommendation techniques as well as One-Class data techniques. We compare each of these techniques and combine them together to make an ensemble of recommendation techniques. We will also discuss how we used different Hadoop technologies to filter our data and give a full analysis of how we used logistic regression to find if a store is a appropriate for a mall**

## I. Introduction

It is of great interest for shopping mall owners to determine what store should be selected to fit an empty slot in a (target) shopping mall so that the shopping mall realizes maximum profit from this decision. We have a dataset of all the malls in the USA along with the county information of the malls [1], and all the stores of various categories within each mall. In addition, each mall has a count of the amount of different stores that fall into a certain category. Additionally, we webscraped store data from Yelp.com and the data falls into three groups: category (restaurant, appeals-to-parents, clothing, etc), average rating of all users and how high-ended each store in a category is (Armani, Louis Vuitton are high-ended stores in apparels, we can consider fast-food joints as low-ended stores and fine-dining is high-ended and so on). For example, a mall in a fancy neighborhood in Manhattan might want to fill an empty slot with a Michael Kors outlet, while a mall in a modest neighborhood in New Mexico might make more profit by renting out the space to a Tex-Mex food chain instead. Intuitively, the store that needs to fill in the empty slot will depend on various factors.

To take into account of these factors, we used a series of recommendation system techniques to predict whether a store is suitable for a mall. The malls and stores can be seen as users and items in the context of the current recommendation system literature. We will apply One-Class User-based Collaborative Filtering (OCUCF) [2], Content-Based Recommender Systems [3] along with Weighted Alternating Least Squares (WALS) [4] with various features to recommend stores with malls. Inherently, these models have their own strengths and weaknesses [3]. We combined some of these models together using Ordinary Least Squares as a ensemble technique.

We will also discuss the performance of each these models using Root Mean-Squared Error (RMSE) and Mean Average Precision (MAP) to compare all of these models.

| | HP1 | HP2 | HP3 | TW | SW1 | SW2 | SW3 |
|---|---|---|---|---|---|---|---|
| $A$ | 4 | | | 5 | 1 | | |
| $B$ | 5 | 5 | 4 | | | | |
| $C$ | | | | 2 | 4 | 5 | |
| $D$ | | 3 | | | | | 3 |

Fig. 1. User HP1, HP2, HP3, .... rating items A, B, C and D. Some entries of the matrix are missing

## II. Related Work

This work is an extension of the work done by Joe Jean, a NYU Abu Dhabi student[5]. He webscraped the store information from MallsInfo.com to determine the stores in every mall. [5] also used the pairwise Haversine distance [6] between malls to predict the number of stores of a certain category should be at a mall. We extend upon this by actually predicting an appropriate store for a mall and using different features.

[3] gives a survey of the underlying basis of recommendation systems. Particularly, [3] discusses the goals of traditional recommendation systems is to impute the missing values of a user-item matrix, $X$, where the $ijth$ entry of the matrix represents the amount of utility that user $u_j$ has for item $i_i$. Figure 1 illustrates an example of recommendation systems with missing entries. Recommendation systems use information from filled entries to extrapolate values of missing entries. In particular, [3] discusses about content-based recommendations (CBR) and collaborative filtering (CF).

[3] mentions that CBR makes recommendations to users by using a similarity function $f$ to compute the relationship between users and items. Then, for each user $u_j$, the k-items that are most similar to the user are the top-k recommended items for $u_j$.

To make the comparison between user $u_j$ and item $i_i$, they need to exactly have the same features. In other words, each component of the vectors should represent the same feature. [3] discusses that it can be difficult to find these features. However, the advantage of using this technique is that it does not need data from other users and experimentally, has been able to recommend users new and unpopular items and does relatively well to predict users with unique tastes.

[3] also mentions about CF techniques, which can be broken down into User-Based Collaborative Filtering (UBCF) and Item-Based Collaborative Filtering (IBCF). To formalize

the idea of UBCF, it is a system that makes new predictions by first finding users with similar ratings to other user and then computes the weighted average of their ratings. This is essentially done with an arbitrary metric that relates the features of user x and user y. More formally, let $U = (u_{jf})$ be the matrix that refers to all the user's features where $(u_{jf})$ represents the f feature of user j, $S = (s_{ij})$ : be the similarity matrix between user $i$ and $j$, and $X = (x_{ij})$ : be the item-user rating matrix. Notice that $S$ is computed by a similarity function f, where $s_{ij} = f(U_{i.}, U_{j.})$.

$$X_{ij} = \overline{X_{.j}} + \frac{\Sigma_{l \in N(j;i)} s_{il}(X_{il} - \overline{X_{.j}})}{\Sigma_{l \in N(j;i)} s_{il}} \tag{1}$$

Note that $\overline{X}_{.j}$ is the average rating user $j$ for all items that the user rated and $N(j;i)$ is the set of users that rated item $i$ and is in the same neighborhood as user $j$. Note that neighborhood can be an arbitrary cluster.

The advantage of using UBCF is that information about items is not required unlike CBR. However, it has a difficult time to recommend users with unique tastes and is generally biased towards popular items.

Similarly, an IBCF predictions are made by first finding items with similar ratings to other items and then computes the weighted average of their ratings. This is defined as:

$$X_{ij} = \overline{X_{i.}} + \frac{\Sigma_{l \in N(i;j)} s_{li}(X_{lj} - \overline{X_{i.}})}{\Sigma_{l \in N(i;j)} s_{li}} \tag{2}$$

Note that $\overline{X}_{i.}$ is the average rating for item $i$ for all the users that rated $i$ and that $N(i;j)$ is the set of items that are that are in the same neighborhood as $i$ and are the items that $j$ rated. The neighborhood set scheme discussed about UBCF applied for items. Also, the pros and cons for (IBCF) is similar to UBCF.

For our dataset, we are not specifically trying to impute missing entries of $X$ since our dataset is a One-Class dataset, a dataset where all entries are binary and all every entries are filled in [4]. Negative examples and missing positive examples are embedded as 0s. In other words, 0s can represent if a user greatly dislikes an item or has not considered having the item. One-Class problems can be seen as more difficult than traditional recommendation problems because of this ambiguity. The goal of One-Class data problems is to change appropriate 0s to 1s.

[4] proposes a solution to this problem by using Matrix Factorization Methods. Factorization techniques are sometimes more favorable than Collaborative Filtering and CBR techniques because they are more effective in reducing RMSE than Collaborative Filtering. More formally, let $X$ be the original rating matrix and $U$ and $V$ be the low-rank approximations of the matrix. The problem is defined as [2]:

$$X \approx UV \tag{3}$$

According to [4], it is important to include weights to compute the approximation of $X$. Weights represents the confidence that some of the values $X_{ij}$ must hold true. We know that a user likes a item since $X_{ij}$ is 1, so we must give it high penalization weight $W_{ij}$ if the prediction $(UV)_{ij}$ is greatly different from $X_{ij}$. Thus, the problem can be defined as:

$$\underset{R}{\text{minimize}} \quad \underset{ij}{\Sigma} W_{ij}(R_{ij} - Xij)^2 \tag{4}$$

$$\underset{U,V}{\text{minimize}} \quad \underset{ij}{\Sigma} W_{ij}((UV)_{ij} - Xij)^2 + \lambda(\|U_{i.}\|_2^2 + \|V_{.j}\|_2^2)$$

A more throughout discussion of how to solve this problem can be seen in [4]. Essentially, an alternating least squares approach can be used to solve for U and V, hence the name Weighted Alternating Least Squares.

Even though the traditional recommendation systems are not necessarily applicable to One-Class data problems, they can be adjusted to solve the problem. CBR can still recommend items to users by computing the similarity between user j and item i where $X_{ij} = 0$. For CBR, the zero entries are recomputed and are imputed as similarity scores between item $i_i$ and user $u_j$. However, it is more difficult to use collaborative filtering in the context of one-class problems because all the missing entries are 0s and all the known entries are 1s. Thus, the known entries can naively impute the 0 entries as 1s for one-class problems. With this approach, all the items are just recommended. However, [2] modified the problem by recomputing $X_{ij}$ by the average binary rating of all users with a similarity metric. [2] received relatively good results for doing this. In the context of User-based Collaborative Filtering, the process can be represented as

- Compute $\forall s_{ij} \in S$: $s_{ij} = f(U_{i.}, U_{j.})$
- $\forall\ X_{ij} = 0$
  - Compute $\hat{X_{ij}} = \overline{X_{.j}} + \frac{\Sigma_{l \in N(j)} s_{il}(X_{il} - \overline{X_{.j}}))}{\Sigma_{l \in N(j)} s_{il}}$

  $X_{ij} = \hat{X_{ij}}, \forall\ X_{ij} = 0$

where $N(j)$ is the set of all users that are in the same cluster as user $u_j$.

## III. Design

### A. Data Processing

Even after the data collection process, the data cannot be used directly for analyze and we needed to filter the data and join them together for better analyze by using Pig, Hive and some Python.

Firstly, we needed to join all the columns together from the county demographic data, since they come in 29 different data csv sources. Luckily, the rows of each file were consistent as each row is exactly represented as the same county. It was easy enough to join all of these data together by appending columns. With this new data source, we needed to join the demographic information and the mall information. Since the demographic county information has geographical coordinates of each county and the mall as well has geographic coordinates, the only way to associate the two sources is by joining them by their shortest distance pairwise distance.

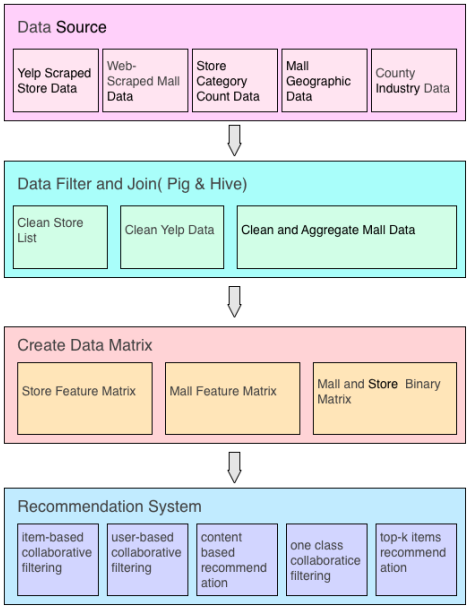


Fig. 2. Design Diagram of Filtering data to do Data Analysis

Secondly, the stores list,a table where each row describes which store was contained in which mall, was collected by [5] and had inconsistencies. Consequently, we needed to process the stores list. In our store list, There are three type of problems, the first one is that many entries in Store field are not actually stores. For example, the stores field would contain names such as ATM, vending machines , community service etc. We needed to delete all of these rows these items. Second, many stores are associated with different names. For example, there are stores with Starbucks and Starbucks Coffee. We need to keep them consistent. Last one is stores have special characters and misspelling and we needed to correct these. In order to get the best prediction result, we create three different regular expression rules to filter our data and create unique id for every store. We wrote Pig Script to load the data first, then use a pig use-define function to filter these data.

Thirdly, after we get the clean store list, we use a yelp crawler to get all there store data. For each unique name of the store, we extracted store name ,category , level of expense,popularity rating. The yelp crawler would have unnecessary columns and we would use Hive to remove these columns. The process of how we filtered our data is shown in the figure and ran our analytic is shown in figure 2.

Because of the modified format of these datasets, they can be easily represented as matrices and for data analysis. Additionally, we converted the store list into a multidimensional dictionary, which is later represented as sparse mall-store matrix.

### *B. Data Analysis with Recommendation Systems*

After formatting the data in certain ways, we used the processed data in various ways to create our recommendation system. Since the $ijth$ entry of the category matrix $V$ represents the amount of $jth$ category stores that the $ith$ mall has, we would like to use this information. Since matrix

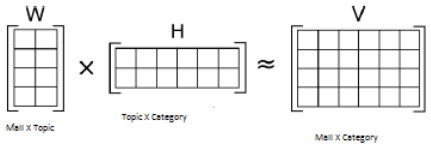


Fig. 3. Nonnegative Matrix Factorization

factorization methods have been used to remove noise from a dataset [7], we applied Non-negative Matrix Factorization (NMF) to obtain a new latent topic representation of the malls. Notice that the optimization problem to find the factors of $V$ is

$$\underset{W,H}{\text{minimize}} \quad \frac{1}{2}\|X - WH\|_2^2 + \lambda(\Sigma_{s_i \in S}\|H_{i.}\|_1^2 + \Sigma_{m_j \in M}\|W_{.j}\|_1^2)$$

Figure 3 illustrates the process of Non-negative Matrix Factorization and the $W$ matrix from the figure are the features that we used as to represent the users for UBCF. We denote this method as "UBCF with NMF-Category". This method relies on the hyper parameters k, the number of topics to split the matrix $V$ by, and $\lambda$, the regularization constant that determines the strength of L1-sparseness. We found appropriate values for these parameters by parameter tuning and cross validation.

We also experimented with other mall features for UBCF. Particularly, we used the demographic information of the malls as their own features. We call this method "UBCF with Demographic Percentage". Additionally, we created another collaborative filter that only integrates the geographic locations of each mall, which we labeled as "UBCF with Geographic Location".

We did create an item-based collaborative filter and filtered the Yelp data to obtain features for the algorithm, but we were not able to finish obtaining results because of time constraints. In the future, we would include this information. We call this technique "IBCF with Yelp Data".

For content based recommendations, we developed a simple approach for items to have the same features as users. For each store, we determined where the store belongs into which mall. For all of these malls, we averaged their demographic information and denote it as the item features. Then, we used cosine similarity to find the similarities of each store to each mall. We call this method "CBR with Demographic Averages". Similarity, we did the same process using the geographic locations of each mall. This will be called "CBR with Geographic Location Averages".

Notice that for the Weighted Alternating Least Squares approach depends on the values of $W$. [4] suggests different ways to set $W$. Particularly, [4] suggested to use $W_{ij} \propto \Sigma_j X_{ij}$. This represents that the more stores that a mall has, they are not going to have any other different stores because they are a successful and large mall. Thus more weight is penalized if their values are different. We call this method "WALS with User Weights".

For our last algorithm, we tried combining the algorithms by using an ensemble. [2] used ordinary least squares to combine their algorithms together by using the predictions of each algorithm as independent variables and the actual values

of the store-mall matrix as independent variables. The only algorithms that we combined are algorithm with id 0-3 in table , mainly because these techniques were relatively fast in giving results and due to time constraints of this project.

For all of the methods that uses a similarity metric, we used cosine distance to compare them except for the ones that use geographic locations, which we used Haversine Distance [6] and modified it to represent similarity. Also, when an algorithm needs neighborhood as input, we just assumed that these are the top 30 similar objects in the neighborhood with respect to an object.

## IV. Result

Traditionally, to test the validity of using a recommendation systems, one can randomly label known values as unknowns and use their algorithm to predict the unknown labels. For our project, we randomly partition the coordinates of the store-mall matrix into 10-different partitions such that each partition has equal distribution zeros and ones. Then, we computed the accuracy of our algorithms with respect to different scores by using cross validation. An illustration of this testing technique is shown in table III

TABLE I. Actual Data

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

TABLE II. Training Data

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

TABLE III. Transforming Data. Accurately transform 0s in training data to the ground truth of 1s

For comparing the results of the other recommendation models, we used two scores, Mean Average Precision and Root Mean Squared Error. Obviously, a model with lower root mean squared error would represent a better model as the prediction matrix produce by the recommendation system roughly approximates the actual matrix.

However, in the literature of One-Class data [4] [2], a score that is more often used is Mean Average Precision, naturally because one-class data problems tend to have low mean-average precision score. Mean Average Precision is derived from Precision. Precision is the percentage of items that are correctly labeled as 1s given the items that the recommendation system predicted as 1s. In other words:

$$Precision \frac{TP}{TP + FP} \tag{5}$$

where TP represents true positive and FP represents false positives, which is the amount of times that the an algorithm predicts as a 1, but it is actually a 0. For ranking items in the context of recommendation problems, the predicted values are sorted in decreasing order and precision for the top-k items for a user is computed by the number of items that are actually 1s in the top-k list over k. We denote precision on a top-k list for a user as $prec_\mu(i)$ and $pref_\mu(i)$ as an indicator variable if a the user has that item. In the literature of information retrieval,

| id | Algorithms | MAP | RMSE |
|---|---|---|---|
| 0 | PopRec | 0.389 | 0.389 |
| 1 | UBCF with NMF-Category | 0.447 | 0.072 |
| 2 | UBCF with Demographic Percentage | 0.379 | 0.0765 |
| 3 | UBCF with Geographic Locations | 0.291 | 0.079 |
| 4 | CBR with Demographic Averages | 0.0338 | 0.94 |
| 5 | CBR with Geographic Location Averages | 0.006 | 3261 |
| 6 | WALS with User Weights | 0.378 | 0.083 |
| 7 | Ensembler: 0+1+2+3 | 0.412 | 0.071 |
| 8 | IBCF with Yelp Data | TBA | TBA |

TABLE IV. Results of Recommendation Systems

Average Precision is defined as:

$$AP_\mu = \frac{\Sigma_{i=1}^{N} prec_\mu(i) \times pref_\mu(i)}{\#\text{ of preferred items}} \tag{6}$$

Simply, Mean Average Precision summarize rankings from multiple queries by averaging average precision scores of all users.

Figure **??** shows the scores on 10,000 stores for all the malls. The MAP scores for these algorithms are quite fairly high compared to other works that dealt with this problem [2] where the highest MAP score gained was $5\%$. This may be due to the fact that the store-mall matrix is not very sparse and big compared to other datasets. Notice that the baseline technique to compare all of these techniques is PopRec. PopRec simply recommends the most popular stores to malls given that the mall does not have that store. PopRec performs quite well since the most shopping malls are consisted of popular stores/chains that are dominating the markets. Some recommendation systems, such as "CBR with Geographic Location Averages" perform fairly bad because they do not recommend stores that are not popular, but rather by some other criterion.

Also, it is interesting to note that "WALS with User Weights" is not the highest performing algorithm even though WALS techniques are designed for One-Class data problems. The WALS algorithm requires a series of multiplied matrices that results into a positive definite matrix to be inverted, which also includes the matrix $X$. If $X$ happens to have a row or a column that only consist of 0s, then that matrix has an eigenvalue of 0 and therefore cannot be invertible and leads to computation errors. During Cross Validation, it made some rows of $X$ to consist of only 0s since there are stores that appeared in one store. To overcome this issue, we let $X = X + \alpha$, where $\alpha$ is a small scalar. Doing this can lead to rounding off errors. Despite the performance of WALS, [2] had similar results as well as their collaborative filtering technique outperforms WALS.

Another observation to point out is that the "Ensembler" is not the highest performing algorithm in dataset. This is due to the fact that Ordinary Least Squares is very susceptible to overfitting and algorithm memorized ensemble memorized the data. This is highly due to the fact that Ordinary Least Squares was used rather than a more profound boosting technique, such Gradient Boosting [8], which has been shown to be successful for combining algorithms together.

## V. Conclusion

We discussed how we recommended stores to malls. A part of this paper focuses how we used Hadoop technologies

to filter our data and used logistic regression to recommend a small sample of store to all malls in the dataset. Another component of the paper is to use various recommendation algorithms to recommend stores to malls.

In the future, instead of using county information, it would be better to use more specific information, such as town information, to represent the demographic information of a shopping mall. This should lead to better results since towns within counties can be polar opposites of each other. Also, in the future, it would be interesting to evaluate the performance of item-based collaborative filtering on this dataset as well implementing gradient boost, to obtain better results for our dataset. Additionally, a better method of cross validation has to be used to overcome the problems presented in WALS. Other model selection models in determining the validity of recommendation systems, such as Akaike Information Criterion, may be worth exploring as well.

## VI. Acknowledgment

We would like to thank Roy Lowrance and Dennis Shasha for giving us advice for this project as well as the opportunity to continue upon the work of this mall dataset. We would also like to thank Suzanne McIntosh for giving us advice on helping us decide which Hadoop technologies we should run and the opportunity to use Amazon Web Services for running our dataset and algorithms on a cluster.

## References

[1] B. of Labor and Statistics, "Quarterly census of employment and wages," 2013.

[2] Y. Li, J. Hu, C. Zhai, and Y. Chen, "Improving one-class collaborative filtering by incorporating rich user information," in *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, 2010, pp. 959–968.

[3] A. Rajaraman and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press, 2011.

[4] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang, "One-class collaborative filtering," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 2008, pp. 502–511.

[5] J. Jean, R. Lowrance, and D. Shasha, "Store-mall recommendation," http://ml2014.herokuapp.com/, accessed: 2015-02-01.

[6] "Haversine distance," accessed: 2015-03-01.

[7] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, 2001, pp. 556–562.

[8] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.