Recommender System for a B2B Online Ordering

Jaume Puigbò Sanvisens
Universitat de Barcelona
Barcelona, Spain
jpuigbo88@gmail.com

Abstract

In this paper we build a recommender system for a B2B company supplying office products through distributors which has lots of SKUs grouped into families. For this reason, the recommended items will be families. The goal of the system is to entice the customers to click and perhaps buy products of families which they do not presently buy on the online system.

1 Introduction

Recommender systems nowadays are very present in online shopping web pages. It is a good a way to show new products unknown to the users. If you add some intelligence by giving them the products they may be more interested in, then it increases the possibility that they will add them to the shopping cart, which will give you more sales and more profits.

The idea of giving intelligence to the recommender system is based in the hypothesis that similar users buy more or less the same products. This hypothesis is used to develop the user-based collaborative filtering. On the other hand, the hypothesis of a item-based recommender system is that a user that is interested on a product maybe is also interested in other similar products. In the section 3 we will explain two different ways to compute the similarity between users or items.

The objective of this work is to implement a recommender system for a B2B online ordering of a company specialized in office products. However this company works with a lot of different SKUs grouped into families. For this reason the objective of this recommender system is to suggest families of products instead products themselves.

In the next section we show which features the data have and some properties. In section 4 we will find which strategy we follow to determine which is the best model to implement. To make this decision we will evaluate the results in two ways, the mean absolute error and the Kullback-Leiber divergence.

2 About the data

This Section will present the reader an overview of the data and some properties of the sales of this company.

2.1 Data exploration

The data provided by the company consists of the orders made by the B2B system of approximately one year. The company has sold 2908 different SKUs to 1905 customers in 66888 orders during this time. The company has currently about 167 different families. Remember that we will finally recommend families.

Table 1: Data description

Number of customers	1905
Number of products	2908
Number of orders	66888
Number of families	167

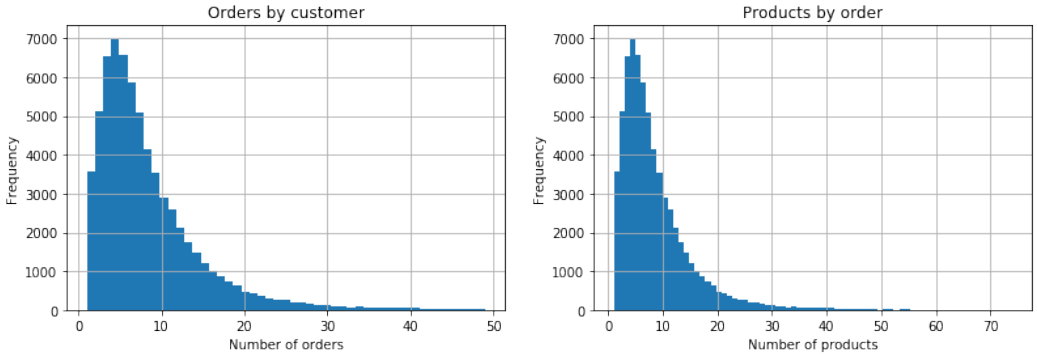

Figure 1: Number of orders per customer (left). Number of products per order (right).

The histograms of Figure 1. show the number of products per order on the right and the number of orders per customer on the left in the course of the year.

It seems that the majority of customers buy more than once on a year and, furthermore, they buy more than one item per order.

In the Figure 2. we can observe the number of products purchased by customer on the left. On the right we have the same result filtering with the customers that purchased less than 50 products.

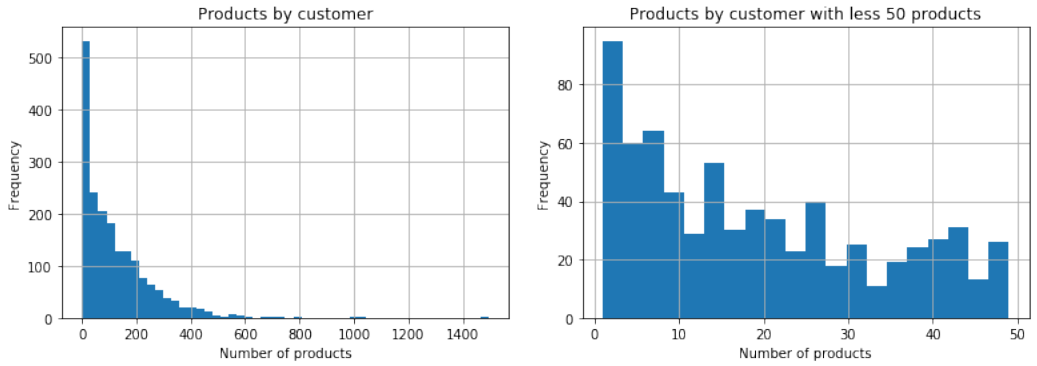

Figure 2: Number of products purchased per customer (left). Number of products purchased per customer that purchased less than 50 products (right).

We can appreciate that there are customers with less that five products purchased. In this case recommendations are very difficult because one can consider that there is not enough data. We will suggest the top families for this case.

And the Figure 3. is a histogram of the number of products per family, where we can appreciate that most of families have less than 30 products and some families have more than 70 products.

3 The model

In this section we will explain each part of the model of the recommender system. Firstly, we will see the construction of the user-item matrix and how to split it in training and testing data. Then we explain the most important part of the model, how to compute the similarity matrix. We will present

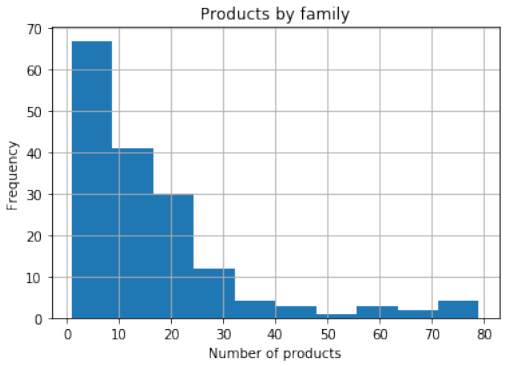

Figure 3: Number of products per family

two different methods to compute the similarity and two kind of similarity matrices for each method, one for user-based model and another for item-based model. Finally we introduce the predict function that we have used.

3.1 User-item matrix

We construct a matrix where the rows are user and the columns are items. Then we compute the frequency that a user has purchased all the items and we normalize this row by the maximum frequency item. Then we add all families as items and we compute for each family its rate, the sum of the rates of the products of this family. After these operations we have constructed a matrix R of dimensions $M \times (N + P)$, where M is the number of users, N is the number of items and P is the number of families. Note that each row of R sums 2.

We have divided the training and testing sets randomly, so that the training set has approximately the 80% of the data and the testing set has approximately 20% of the data.

3.2 Similarity

The similarity can be seen as training of the system, therefore we compute the similarity using only the training set.

3.2.1 Collaborative user-based

To compute the similarity between users, we used the Pearson correlation:

$$sim(u,v) = \frac{\sum_{p \in P}(r_{u,p} - \overline{r}_u)(r_{v,p} - \overline{r}_v)}{\sqrt{\sum_{p \in P}(r_{u,p} - \overline{r}_u)^2}\sqrt{\sum_{p \in P}(r_{v,p} - \overline{r}_v)^2}},$$

where P is the set of common items rated by the user u and the user v, $r_{u,p}$ is the rating of the item p by the user u and $\overline{r}_u$ is the mean rating given by the user u.

Observe that the similarity computed using the Pearson correlation is symmetric

$$sim(u,v) = sim(v,u).$$

Therefore we only compute a triangular part of the similarity matrix without the diagonal. It means that we must do $\frac{M^2-M}{2}$ computations to compute the similarity matrix between users.

3.2.2 Collaborative item-based

The process to compute the similarity between items is analog to the user-based process. We used the Pearson correlation to compute the similarity between items:

$$sim(i,j) = \frac{\sum_{p \in P}(r_{p,i} - \overline{r}_i)(r_{p,j} - \overline{r}_j)}{\sqrt{\sum_{p \in P}(r_{p,i} - \overline{r}_i)^2}\sqrt{\sum_{p \in P}(r_{p,j} - \overline{r}_j)^2}},$$

where P is the set of common users that rated the items i and j, $r_{p,i}$ is the rating of the item i by the user p and $\overline{r}_i$ is the mean rating of the item i.

As in the case of the user-based the similarity between items is symmetric, that is we need $\frac{(N+P)^2-(N+P)}{2}$ computations to compute the similarity matrix between users. Remember that we consider the families as products.

3.2.3 Graph-based

It is possible to use structural measures on the user-item graph, rather than the Pearson correlation coefficient, for defining neighborhoods. Such an approach is more effective for sparse ratings matrices because one can use structural transitivity of edges for the recommendation process and this is our case.

The user-item graph is defined as an undirected and bipartite graph $G = (N_u \cup N_i, (A, w))$, where N_u is the set of nodes representing users, and N_i is the set of nodes representing items. An undirected edge exists in A between a user and an item, if and only if user has rated the item and the weight of this edge is the rate of the user to the item. Observe that all edges in the graph exist only between users and items (see Figure 4. left).

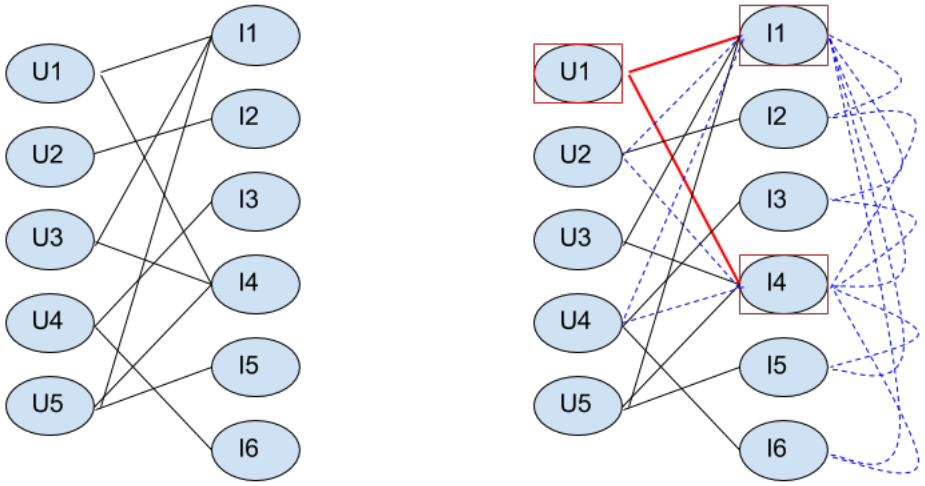

Figure 4: Graph example.

The main advantage of graph-based methods is that two users do not need to have rated many of the same items to be considered neighbors as long as many short paths exist between the two users. Therefore, this definition allows the construction of neighborhoods with the notion of indirect connectivity between nodes. Of course, if two users have rated many common items, then such a definition will also consider them close neighbors. Therefore, the graph-based approach provides a different way of defining neighborhoods, which can be useful in sparse settings.

The neighborhood of a user is defined by the set of users that are encountered frequently in a random walk starting at that user. To compute the similarity of a user with the other users, we add edges between the items that this user rated and all other nodes with a weight of $\frac{1}{n}$, where n is the number of nodes (see Figure 4. right). Then we compute the PageRank of this graph to simulate the random

walk and we obtain the similarity of this user with all the others. Repeating the process for all users we obtain a similarity matrix between users.

On the other hand, if we do the analogous process adding edges between the users that rated a certain item and repeating the process for all items, we obtain a similarity matrix between items.

3.3 Predict

We need to predict the rating of a item or family. To compute the rating given a user and item we use the following formula

$$r_{u,i} = \frac{\sum_{v \in P_u(i)} r_{v,i} sim(u,v)}{\sum_{v \in P_u(i)} sim(u,v)},$$

where $P_u(i)$ is the set of the top-k similar users to the user u that rated the item i.

Note that this formula is to be applied to a user-user similarity matrix. The formula for a item-based method is obtained exchanging the role between user and item as follows

$$r_{u,i} = \frac{\sum_{j \in P_i(u)} r_{v,i} sim(i,j)}{\sum_{j \in P_i(u)} sim(i,j)},$$

where $P_i(u)$ is the set of the top-k similar items to the item i that have been rated for the user u.

Note that k is a very important parameter. We will test different values for k to determine which is the best for each method. We will see these results in detail in the next section.

There are other possible predict functions that solve some problems based on that the rating is subjective. In other words, users have a different way to rate items that they like or dislike, but this is not our case since the scores are set by the system objectively.

Finally, the list of the families that the system will recommend for each user are the families that have a higher rate and the user has never purchased any item of this family.

4 Evaluation

In this section we will present the results obtained by evaluating the different methods explained in the previous section. A first evaluation will be the mean absolute error on the values of the test set, which we will use to determine the number of similar users or items that we will use to make the prediction. The second evaluation will be the Kullback-Leiber divergence to compute the difference between the families rate distributions. Finally, we will suggest an A/B testing to have a online measure.

4.1 Mean Absolute Error (MAE)

Remember that we had a training set to compute the similarity matrix and a test set that we will use to compute the mean absolute error. The mean absolute error (MAE) is defined as

$$MAE = \frac{\sum_{i \in T} |r_i - \hat{r}_i|}{n},$$

where T is the test set, r_i is the rating in the test set, $\hat{r}_i$ is the predicted rate and n is the number of elements in the test set.

We will use this evaluation as a first evaluation between methods, but also to determine the k to use in the predict function. Remember that in the the predict function we use the set $P_i(u)$, that is the set top-k similar users or items depending on the algorithm we run.

The Table 2. shows the results obtained by evaluating the mean absolute error in each method with $k = 5, 10, 15, 20$. As we can observe the best result of the user-based method and both PageRank

Table 2: MAE evaluation

Method	MAE (k=5)	MAE (k=10)	MAE (k=15)	MAE (k=20)
User-based	0.0187	0.0142	0.0123	**0.0113**
Item-based	**0.00425**	0.00440	0.00449	0.00454
PageRank (user-based)	0.00968	0.00962	0.00962	**0.00961**
PageRank (item-based)	0.0117	0.0105	0.00989	**0.00957**

methods is with $k = 20$, and $k = 5$ in the case of the item-based method. In this case the best result is the item-based method with $MAE = 0.00425$. It is a very good result and a big improvement compared to the other methods.

4.2 Kullback-Leiber Divergence

The second metric that we use to evaluate and compare our models is the Kullback-Leiber divergence. The Kullback-Leiber divergence from Q to P is defined as

$$KL(P||Q) = \sum_i P(i) \ln \frac{P(i)}{Q(i)},$$

where P and Q are probability distributions of discrete random variables. Note that $KL(P||Q)$ is only defined if $Q(i) > 0$ for all i such that $P(i) > 0$. In other words, it is the expectation of the logarithmic difference between the probabilities P and Q, where the expectation is taken using the probabilities P.

We can see also the Kullback-Leiber divergence like a distance between two probability distributions given that $KL(P||Q)$ is always positive and $KL(P||Q) = 0$ if and only if $P = Q$.

In our case we take P as the real ratings of all families for each user and Q is the normalized predicted family ratings computed using one of the methods.

Observe that if the distribution of the real ratings of families and the distribution of the predicted ones are similar it means that we have made a good prediction. Therefore, the lower Kullback-Leiber divergence the better prediction for the families. Kullback-Leiber evaluation is a way of measuring that recommendations are really relevant and that irrelevant recommendations are omitted. In a recommender system is very important that we don't give bad recommendations, because they can annoy the user. Therefore we don't want high values of the Kullback-Leiber evaluation.

In Figure 5. we have an histogram of the results of the Kullback-Leiber divergence for each method on the left and a boxplot on the right. Clearly in this case the best model is the user-based. It has a lower mean, a lower standard deviation and the outliers are not so big, which means not so bad recommendations. Therefore we suggest to implement the user-based model in the system.

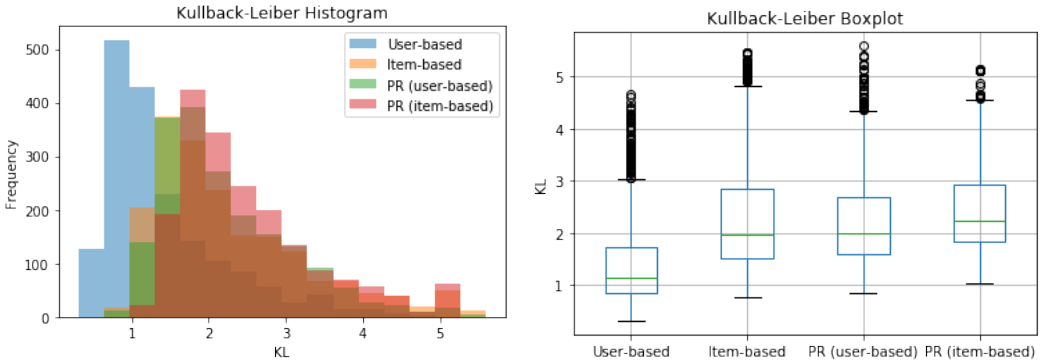

Figure 5: Kullback-Leiber histogram (left). Kullback-Leiber boxplot (right).

In Figure 6. we can see some examples of the difference between the real distribution of the families rates and the predicted distribution. The Kullback-Leiber divergences are between 1.3 and 1.5, that is approximately the mean of the value of the Kullback-Leiber divergence of the user-based method. We can see that the difference between the real and the predict distribution is not so big. Then it seems that the recommender system works well. On the other hand in Figure 7. we find some examples of the the real and predicted distributions of the families such that the Kullback-Leiber divergence is more than 4. These are the outlier cases and it seems that these cases are for users that have less than five ratings. For these users that have less than five ratings, the system will recommend the most popular families.

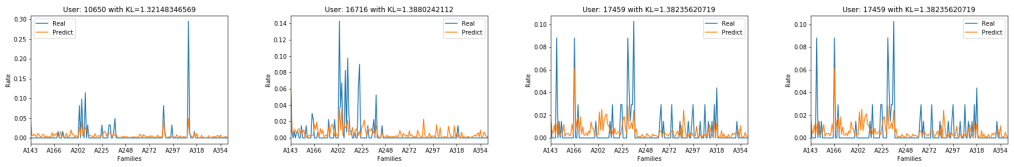

Figure 6: Examples real distribution versus predicted distribution with $1.2 < KL(P||Q) < 1.4$.

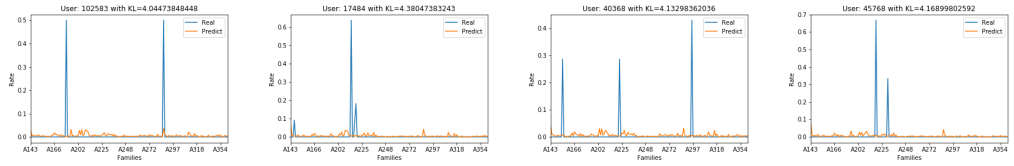

Figure 7: Examples real distribution versus predicted distribution with $KL(P||Q) > 4$ (outliers).

4.3 A/B Testing

The previous evaluations are offline metrics to decide witch is the best model. The problem of these evaluations is that there are not realistic metrics of our problem. We suggest that the company do an A/B testing to evaluate if our model works well.

Nowadays, the company recommends some families randomly. Then we can make an A/B testing where a set of users has the random recommendations and another has the recommendations predicted by our model. We suggest two different metrics. We can track the number of clicks on the buttons of the website, where the recommendation families appear and we can track also the number of times that the users add some item in the shopping basket after clicking a recommended family. We have to take care how to select the users of the group A and B, this is a critical point. We have to select it randomly, but afterwards we must be sure that the groups are balanced. It means, for example, it can't happen that one group has the best customers and the other the smaller ones.

In this way we will have an online and realistic metric that our recommender system works well or not.

5 Conclusions

This paper presented the election of different models to make a recommender system for a B2B system of a company specialized on office products. We saw four different implementations: user-based, item-based, PageRank (user-based) and PageRank (item-based). The evaluation to decide which is the best model is using the Kullback-Leiber divergence. The winner method is the user-based with a clear advantage over its competitors. After the implementation it will be great to make an A/B test to make sure that this recommender system gets better results versus a random one.

Code

You can find the code related to this paper in Github (https://github.com/jpuigbo88/Recommender-System-B2B.git).

References

[1] Charu C. Aggarwal, Recommender System: The Textbook, 2016, Springer

[2] Kullback–Leibler divergence, Wikipedia, https://en.wikipedia.org/wiki/Kullback-Leibler_divergence

[3] Jones E, Oliphant E, Peterson P, et al. SciPy: Open Source Scientific Tools for Python, 2001, http://www.scipy.org/

[4] Pearson Correlation, Scipy, http://www.statsoft.com/textbook/glosp.html#Pearson%20Correlation

[5] Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering, 13, 22-30 (2011)

[6] John D. Hunter. Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95 (2007)

[7] Wes McKinney. Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56 (2010)

[8] Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart, "Exploring network structure, dynamics, and function using NetworkX", in Proceedings of the 7th Python in Science Conference (SciPy2008), Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11–15, Aug 2008

[9] Fernando Pérez, Brian E. Granger, IPython: A System for Interactive Scientific Computing, Computing in Science and Engineering, vol. 9, no. 3, pp. 21-29, May/June 2007, doi:10.1109/MCSE.2007.53. URL: http://ipython.org