RecSys Challenge 2015

Shanmukha Bharat Vakalapudi

May 10, 2015

Abstract

Given a large set of click and buy events during a typical session in an e-commerce website, how can we model the click and buy events? How can we predict if a click event from a session results in a buy event? I use a model that models the click and buy events using information like sessionId, ItemId, category and the timestamp [7]. I also propose a model that would use the information like timestamp from the session, category of the item, number of clicks per item, etc. to predict if a click event from the session results in a buy event. I have used these models on the data from RecSys Challenge 2015 and predicted some buy events given a set of click events.

1 Introduction

In recent years, we have witnessed a tremendous growth of websites in terms of number and usage. This results in accumulation of huge amounts of data per each website. This data can be of two types.

1. **Implicit Data** [11]: This type of data doesn't explicitly show the user's interests. User's interests are inferred by observing the user's actions. Example : Click-buy events from an e-commerce website.

2. **Explicit Data** [12]: This type of data explicitly shows the user's interests. Example : Movie or Hotel rating data.

Lot of research is being done addressing these types of data and making use of it. This is where recommender systems come into play. Recommender systems is one of the ways we can make use of the data and make some useful predictions. In this work, I have used a simple approach to build a recommender system, using 6 months of data(implicit data) from an e-commerce website. The data represents six months of activities of a big e-commerce businesses in Europe selling all kinds of stuff such as garden tools, toys, clothes, electronics and much more. The recommender system will do two tasks.

1. Predict if a user is going to buy in a particular session and

2. If he buys, what would be the items he is going to buy

To build this recommender system, I have used Apache Mahout's Java based library(mahout-core-0.9) to model the click and buy events from the data. I have also proposed a model to do the two tasks mentioned above. Generation of this model can be explained in three steps.

1. Analyse the implicit data and gather useful information from it.
2. Eliminate the user-item pairs which mayn't be bought. This is done using the information from first step.
3. Use the remaining data and generate a model, that would predict the items user is going to buy.

A more detailed explanation will be given in the later sections of this paper. I demonstrate the application of this model on the dataset and will discuss the results. I will also explain the experiments I have done on the model.

2 Problem Statement

[10] In this year's edition of the RecSys Challenge, YOOCHOOSE is providing a collection of sequences of click events; click sessions. For some of the sessions, there are also buying events. Hence, given a sequence of click events performed by some user during a typical session in an e-commerce website, the goal is to

1. Predict if a user is going to buy something or not and
2. If he buys, what would be the items he is going to buy

2.1 Notations

The terms 'user' and 'session' mean the same thing in this paper. This is because I am using each session as a user in my model.

3 Literature Review

Recommender systems have become an important research area since the emergence of the first research paper on collaborative filtering in the mid-1990s. The article [1] shows the amount and the significance of the research done for about a decade. As the paper describes, the recommender systems can be classified into Collaborative filtering and Content based filtering techniques. As we go into Collaborative filtering, there is a lot of research going on how to integrate the data mining techniques for better recommender systems. There are papers like [13][14] explaining the different collaborative filtering techniques. Out of these techniques, I thought user-user collaborative filtering and the probabilistic matrix factorization are more convincing for my problem statement.

User - user collaborative filtering, also known as k-NN collaborative filtering, was the first of the automated CF methods. In K-NN, find other users whose

past click-buy behavior is similar to that of the current user and use their behaviour to predict what the current user will buy. The probabilistic methods generally aim to build probabilistic models of user behavior and use those models to predict future behavior. The core idea of probabilistic methods is to compute the probability that user 'u' will purchase an item i. These probabilistic methods are combined with SVD and other factorisation techniques of data mining to form more complex methods. Probabilistic matrix factorisation is one of these methods. The article [1] suggests, majority of the work in the shopping domain is done using K-NN and clustering.

Clustering techniques work by identifying groups of users who appear to have similar preferences. Once the clusters are created, predictions for an individual can be made by averaging the opinions of the other users in that cluster. Some clustering techniques represent each users with partial participation in several clusters. The prediction is then an average across the clusters, weighted by degree of participation. Clustering techniques usually produce less-personal recommendations than other methods, and in some cases, the clusters have worse accuracy than nearest neighbor algorithms [3].

[2] present a detailed taxonomy and examples of recommender systems used in E-commerce and how they can provide one-to-one personalization and at the same can capture customer loyalty. Although these systems have been successful in the past, their widespread use has exposed some of their limitations such as the problems of sparsity in the data set, problems associated with high dimensionality and so on. Sparsity problem in recommender system has been addressed in [4][5].

My work tries to apply user-user collaborative filtering on E-commerce implicit data. I may face the sparsity and the scalability problems but, since this is a research and learning oriented project I am willing to face the extent of these problems.

4 Methods and Techniques

Before I go into detail about the methods and techniques I have used, I will give a brief introduction to the library I have used.

About the library : I have used apache mahout's "mahout-core-0.9" Java API [6] to implement this project. This API has a good set of algorithms that solve the problems on machine learning, recommender systems, classification and clustering. This API has a class called FileDataModel [7] which would take a '.csv' file as an input and builds a data model from the data. Given a record/line, the fields should be in the order userId, ItemId, preference, timestamp. Hence, for the model to work on my dataset, I am using category values as preference values for yoochoose-clicks.csv and quantity as preference value for yoochoose-buys.csv.

Methods : The generic idea is that recommender systems apply data analysis techniques to the problem of predicting the items users would like to purchase

at E-Commerce sites, by producing a predicted likeliness score. I have changed the generic idea a bit to solve my problem. For the user based collaborative filtering to work, I am considering sessions from the data as users and I am using information from the data analysis to develop my prediction model. The process is as follows:

1. **Analysing the implicit data :** In this step, I have used the data files to gather as much information as I can through data analysis. The results of this step are presented in 5.1.1.

2. **Eliminate the user-item pairs which mayn't be bought :** In this step, I have changed my data to fit the requirements of FileDataModel class and built the models for all the input data. Using these input data models and the information from section 5.1.1, I have defined a base condition to identify the user-item pairs which may not result in buy events. The snapshot of base condition is as follows:

```
if(!(day == "Tuesday" && month != "08" && num_of_clicks < 2
                    && !topItems.contains(each) && unique_items_clicked < 2)){
```

Figure 1: Eliminating the user-item pairs

To explain, If the day is Tuesday and month is not august and number of clicks on that item is less than 2 and if the item is not in the list of most bought items and number of unique items clicked by that user are less than 2, then remove the user-item pair from the list of possible buy events.

3. **Final Model :** After I have computed the possible buys, I have used two algorithms to get the final predictions. They are as below:

   (a) This is the algorithm for making the predictions.

Inputs : We have click and buy events from the training data. We also take the output from step-2 as clicks of test-data.

Algorithm :

Step-1: Calculate the number of times a user has clicked an item. Do this for all the users in clicks of training set and clicks of test set.
Step-2: For each user-item pair in clicks of test data
- Step-2.1: Get the number of times user has clicked this item.
- Step-2.2: If the number of clicks > 1
  - Step-2.2.1: Find the users from clicks of training data, who have clicked the same item same number of items. Consider all these users as similar to the current user.
  - Step-2.2.2: Take all the users similar to the current user and find the similarity measure with the current user using computeSimilarity function.
  - Step-2.2.3: Take the user with maximum similarity measure and check if he has a buy event with this item.
  - Step-2.2.4: If the most similar user has a buying event, then take the current user and the item as a buy event. Add the user to the predicted users. Else,
    - Step2.2.4.1: if (((the month of click event is August or April or May) and (the hours is between 8 and 11 or 18 and 19) and (if the list of most bought items contain this item)) and (number of clicks on this item is > 3) and (if the user is present in predicted users list)), then take the current user and item as buy event.
- Step-2.3: Else,
  - Step-2.3.1: if number of unique items clicked is equal to 1
    - Step-2.3.1.1: if (((the month of click event is August or April or May) and (the hours is between 8 and 11 or 18 and 19) and (if the list of most bought items contain this item)) and (number of clicks on this item is > 3)), then take the current user and item as buy event.
  - Step-2.3.2: Else,
    - Step-2.3.2.1: if (((the month of click event is August or April or May) and (the hours is between 8 and 11 or 18 and 19) and (if the list of most bought items contain this item)) and (number of clicks on this item is > 3) and (if the user is present in predicted users list)), then take the current user and item as buy event.

Output : Final predicted buy events for a given set of click events.

Figure 2: Final prediction model

(b) This is the algorithm for finding the similarity measure.

```
Inputs: User from the test data, One of the similar users of the current user.

Algorithm:  userSimilarity(user1,user2)

Step-1: Get all the items and corresponding timestamps, user1 has clicked.
Step-2: Get all the items and corresponding timestamps, user2 has clicked
Step-3: For each item and timestamp from Step-1,
    Step-3.1: For each item and timestamp from Step-2,
        Step-3.1.1: If the category is same for both the items similarity is 1, else 0.
        Step-3.1.2: If the month from the timestamp is the same similarity is 1, else the similarity is given as
            similarity(month) = 1-abs(m1-m2/12)
        Step-3.1.3: If the day from the timestamp is the same similarity is 1, else the similarity is given as
            similarity(day) = 1-abs(d1-d2/30)
        Step-3.1.4: If the hours from the timestamp is the same similarity is 1, else the similarity is given as
            similarity(hours) = 1-abs(h1-h2/24)
        Step-3.1.5: If the minutes from the timestamp is the same similarity is 1, else the similarity is given as
            similarity(min) = 1-abs(m1-m2/60)
        Step-3.1.6: Add the similarities from the above five steps and divide it by 5, to get the average similarity per item. Store these values in a data structure.
Step-4: Get the final average of all the values stored in the data structure. This is the required similarity measure between the two users.

Output: A value showing the measure of similarity between two users.
```

Figure 3: Computing the similarity values

5 Discussion and Results

5.1 Datasets

I have used the dataset provided by the RecSys Challenge 2015. The YOOCHOOSE dataset contains a collection of sessions from a retailer, where each session is encapsulating the click events that the user performed in the session. For some of the sessions, there are also buy events; means that the session ended with the user bought something from the web shop. The data was collected during 6 months in the year of 2014, reflecting the clicks and purchases performed by the users of an on-line retailer in Europe. The data files are as below:

1. **yoochoose-clicks.csv :** This file contains a sequence of click events; click sessions. Each click event has 4 attributes. They are sessionId, timestamp, itemId and category. For more information on the attributes, please refer to [10] The basic statistics for this file are as below:
   (a) Total number of click events are 33003944.
   (b) Number of unique sessions in this clicks file are 9249729.
   (c) Number of unique items are 52739.
2. **yoochoose-buys.csv :** This file comprises the buy events of the users over the items. Each record/line in the file has 5 fields. sessionId, timestamp, itemId, price, quantity. For more information on the attributes, please refer to [10] The basic statistics for this file are as below:
   (a) Number of items bought are 1150753.

(b) Number of sessions with buys are 509696.

(c) Number of unique items bought are 19949.

3. **yoochoose-test.csv :** This file comprises of only clicks of users over items. This file served as a test file in the RecSys challenge 2015. The structure is identical to the file yoochoose-clicks.dat but you will not find the corresponding buying events to these sessions in the yoochoose-buys.dat file.

5.1.1 More information and statistics

1. All items bought had at least one click in the same session.
2. Small subset of items(1994 of 19949) contains many buys, while most of the items get very less number of buys. I have taken these small subset of items as a list of most bought items.
3. Percentage of sessions with buys is 0.055103.
4. If a user has click events on 2 or 3 unique items, it is highly probable that he will buy at least 1 unique item.
5. Mean and std for the number of clicks per session

mean	3.568098
std	3.787520

6. Mean and std for the number of items bought per session (considering only the buying sessions)

mean	2.257724
std	1.933342

7. Most number of buys happen in August and least number of buys happen in July [8].
8. Least number of buys happen on Tuesday[8].
9. Most buys happen between 8h to 11h and 18h to 19h [8].

5.2 Evaluation Metrics

A lot of research is being done on several types of measures, for evaluating the quality of a recommender system. When we are trying to predict what the user is going to buy or what the user is going to be interested in watching, we are not really interested in the ratings user is going to give. So, we select a test user, hide his/her buy events, and ask the recommender to predict a set of items that the user will buy. We then have four possible outcomes for the predicted and the hidden items as shown below.

	Bought	Not-Bought
Bought	True-positive(tp)	False-negative(fn)
Not-bought	False-positive(fp)	True-negative(tn)

In the above table rows represent the hidden items(ground truth) and the columns represent the predicted values. Using this table we can define the following evaluation measures:

$$Precision = \frac{tp}{(tp + fp)} \tag{1}$$

$$Recall = \frac{tp}{(tp + fn)} \tag{2}$$

I have also evaluated my results using the evaluation measure given by the RecSys challenge page. [10]

5.2.1 Application

1. Precision with respect to this project means number of correct predictions per total number of predictions.
2. Recall with respect to this project means number of correct predictions per number of correct predictions that should have been made.
3. More number of predictions typically improves recall, and decreases the precision.

5.3 Experimental Results

I have tried to experiment by making changes to my prediction model. I have explained each one below:

1. **Initial model :** In my initial prediction model, I have used the entire data from the training and test sets. In this model, I have taken a single user from the test set and tried to find the similarity measure with all the users from the training set i.e I have tried to compute the similarity measure 9249729 times for a single user in the test set. I have described the model below:

Step-1: Find the users from clicks most similar to the user being tested.
Step-2: If these similar users are in the buy events then,
    Step-2.1: Given a set of click events and buy events, find the buys per clicks ratio for all the buy events of the most similar user.
    Step-2.2: If the click event for test user and the click event for most similar user have same ItemId,
        Step-2.2.1: Multiply the ratio from Step-2.1 with the number of clicks from test user.
        Step-2.2.2: If the result is > 0, item is added is added to the final result. Else, predict the items using the information from data analysis.
Step-3: Else, predict the items using the information from data analysis.

Figure 4: Initial model

Because of the huge number of similarities being computed, each test record takes about 15 to 25 minutes to get a final prediction. Also, my prediction model is too weak and may result in a lot of false positives being added to the final result.

Results : For sessionId 5, Max similarity measure is 0.800000011920929 and the predicted items are 214530776. For sessionId 1980, Max similarity measure is 0.7966667413711548 and the predicted items are 214826955.

This is the most basic and the naive model I have written and hence, I have not evaluated the results from my initial model.

2. **Second model :** I have tried to reduce the back drops of my initial model in this model. The changes I have made are :

   (a) I have reduced the number of similarities I compute. To do this, I have computed the number of clicks per item for each user-item pair. I compute the similarities for only those users, who have the same number of clicks for the test item. This has drastically reduced the number of similarities I compute in my main code.

   (b) I have strengthened my prediction model, in order to reduce the number of false positives. The resulting predictive model is described in section 4.

Step-1: Calculate the number of times a user has clicked an item. Do this for all the users in clicks of training set and clicks of test set.
Step-2: For each user-item pair in clicks of test data
Step-2.1: Get the number of times user has clicked this item.
Step-2.2: If the number of clicks > 1
Step-2.2.1: Find the users from clicks of training data, who have clicked the same item same number of items. Consider all these users as similar to the current user.
Step-2.2.2: Take all the users similar to the current user and find the similarity measure with the current user using computeSimilarity function.
Step-2.2.3: Take the user with maximum similarity measure and check if he has a buy event with this item.
Step-2.2.4: If the most similar user has a buying event, then take the current user and the item as a buy event. Else,
Step2.2.4.1: if (((the month of click event is August or April or May) and (the hours is between 8 and 11 or 18 and 19) and (if the list of most bought items contain this item))
or
((if the number of unique items clicked is 1) and (number of clicks on this item is > 3))), then take the current user and item as buy event.
Step-2.3: Else, take the user-item click event and
Step-2.3.1: if (((the month of click event is August or April or May) and (the hours is between 8 and 11 or 18 and 19) and (if the list of most bought items contain this item))
or
((if the number of unique items clicked is 1) and (number of clicks on this item is > 3))), then take the current user and item as buy event.

Figure 5: Second model

Test Data : I have tried testing this model on the original test data. I have run it for more than a day and looking at the rate it is going, I had to stop it. I thought it maybe taking more than 4 days for the entire test set. So, I have taken click and buy events until sessionId 578099 for my testing purpose. I made sure the data is uniform and well spread.

   (a) Training: Number of click events are 1453180. Number of buy events are 46104. Unique sessions between 115619 and 578099 are taken as training sessions.

   (b) Testing : Number of click events are 366095. Unique sessions from 1 to 115619 are taken as testing sessions.

Results :

	Bought	Not-Bought
Bought	3612	2495
Not-bought	33584	-

(a) Actual number of buy sessions are 6107 and the predicted buy sessions are 37196.

(b) Precision is 0.09710722.

(c) Recall is 0.5914524.

(d) RecSys evaluation score is 469.38947

From these results we can say, there are still a lot of false positives which are making the precision value bad. But, the number of correct predictions are actually high which can be seen from the recall value. RecSys evaluation metric doesn't really penalise for predicting wrong items. But, it penalises you for predicting the wrong sessions [9]. This means, we need higher precision for a higher value in RecSys evaluation metric.

3. **Third model :** I have made only minor changes to my second model. I have strengthened the prediction model even more to reduce the false positives.I didn't track the predictions I was making at each step, in the previous model. This resulted in a weak 'else' part of my prediction model. I have made changes to it and the resulting model is the final model. Please refer to figure 2 for the final model.

Test Data : I have used the same data as I have used for evaluating second model.

Results :

	Bought	Not-Bought
Bought	1904	4203
Not-bought	5550	-

(a) Actual number of buy sessions are 6107 and the predicted buy sessions are 7454.

(b) Precision is 0.25543332.

(c) Recall is 0.3117734.

(d) RecSys evaluation score is 938.4887.

From these results, I can say I was able to reduce the number of false positives thereby increasing the precision. But also resulted in decreasing the recall, which is bad. I have got a significant raise in RecSys evaluation score because of the increase in precision. But, I have to find a way to define a model that will balance the precision and recall scores.

6 Conclusion

1. Data analysis plays a major role in building the predictive models. I will have to do even more analysis to get my model right.

2. Selection of the recommender system algorithm depends on the type of data and the statistics you get from the data.

3. This project is a perfect example for showing the pitfalls of user-based collaborative filtering. I have faced data sparsity and scalability issues while building my predictive model.

4. The value of the precision matters when the number of recommendations or predictions you can make are limited. This project doesn't explicitly limit the predictions but, the evaluation measure they are using implicitly states it.

5. The value of recall matters when you don't have any limit on the predictions you make.

6. I have learnt a lot of stuff on how to deal with huge amounts of data and how to analyse the data using R.

7. I have learnt a lot of stuff on Apache mahout's machine learning libraries.

6.1 Directions for Future Work

1. I have to take the amount of time spent per item, for each user-item pair along with number of clicks to find the similar users.

2. I am just considering a single user from the most similar users. There maybe multiple users with the maximum measure. I have to consider all these users to make my prediction.

3. I have to include category as a factor in my prediction model. Category value makes sense because they are having information on special offers and the brands the user prefers.

4. I was having problems dealing with single click events. A bit more information should be gathered from the data about the single click events.

5. I'll have to find the balance between my second model and third model to get better results. This mostly depends on how I handle the single click events.

6. I have to find a way to run my model on the entire test data provided and evaluate my result using the measure they have provided.

7. If i have to do it again, I think probabilistic matrix factorisation is an approach worth trying for this project.

References

[1] Deuk Hee Park, Hyea Kyeong Kim, Il Young Choi, Jae Kyeong Kim, *A literature review and classification of recommender systems research*, available at http://www.sciencedirect.com/science/article/pii/S0957417412002825.

[2] Schafer, J. B., Konstan, J., and Riedl, J. (1999)., *Recommender Systems in E-Commerce. In Proceedings of ACM E-Commerce 1999 conference.*

[3] Breese, J. S., Heckerman, D., and Kadie, C. (1998)., *Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, pp. 43-52.*

[4] Good, N., Schafer, B., Konstan, J., Borchers, A., Sarwar, B., Herlocker, J., and Riedl, J. (1999)., *Combining Collaborative Filtering With Personal Agents for Better Recommendations. In Proceedings of the AAAI-?99 conference, pp 439-446.*

[5] Sarwar, B., M., Konstan, J. A., Borchers, A., Herlocker, J., Miller, B., and Riedl, J. (1998)., *Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System. In Proceedings of CSCW ?98, Seattle, WA.*

[6] *Javadoc for Apache Mahout 0.9 API*, available at http://javadox.com/org.apache.mahout/mahout-core/0.9/overview-summary.html.

[7] *Class FileDataModel- Apache Mahout 0.9*, available athttp://javadox.com/org.apache.mahout/mahout-core/0.9/org/apache/mahout/cf/taste/impl/model/file/FileDataModel.html.

[8] *RecSys Challenge blogs*, available at http://aloneindecember.com/words/recsys-challenge-part-i/.

[9] *RecSys Challenge blogs*, available at https://thierrysilbermann.wordpress.com/2015/01/13/can-we-approximate-the-upper-bound-score-for-the-2015-recsys-challenge/.

[10] *RecSys Challenge page*, available at http://2015.recsyschallenge.com/challenge.html.

[11] *Implicit data*, available at http://whatis.techtarget.com/definition/implicit-data.

[12] *Explicit data*, available at http://whatis.techtarget.com/definition/explicit-data.

[13] Michael D. Ekstrand, John T. Riedl and Joseph A. Konstan, *Collaborative Filtering Recommender Systems*, available at http://files.grouplens.org/papers/FnT%20CF%20Recsys%20Survey.pdf.

[14] Prem Melville and Vikas Sindhwani, *Recommender systems*, available at `http://vikas.sindhwani.org/recommender.pdf`.

[15] *LibRec*, available at `http://www.librec.net/tutorial.html`.

[16] *MyMediaLite*, available at `http://www.mymedialite.net/`.

[17] *libFM*, available at `http://www.mymedialite.net/`.