Market Basket Analysis

Mhyar Kousa1, Szegedi Gabor.2, and Dr. Tomas Horvath3
1 MSc. Data Science, ELTE University - Faculty of Informatics
mhyarov.k@gmail.com
2 PHD. Data Science, ELTE University - Faculty of Informatics
wayasam@gmail.com
3 Head of Data Science and Engineering Department, ELTE University - Faculty of Informatics. 1117 Budapest, Pázmány Péter str. 1/A Northern Building
tomas.horvath@inf.elte.hu
https://www.elte.hu/en/

Abstract. In this report, we use the Decision Tree Classifier as a training model and Random Forest ,Gaussian NB and KNN Classifiers as a base line ,the best accuracy was 94.93 with Random Forest,Gausian NB and Decision Tree to predict frequent products .

Keywords: Decision Tree, Random Forest, Gaussian NB, KNN, AdaLoyal Algorithm, Apriori Algorithm, Most Frequent Products , Matrix Factorization.

1 Introduction

This is report for the project which is carried out as part of Data Science Lab. The topic of project is Shopping Basket. The purpose of the project is to predict the most frequent products inside each basket. Decision Tree Classifier was used for this purpose .

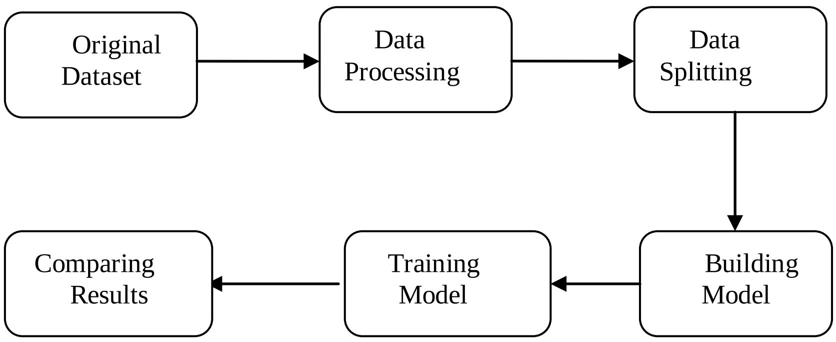

Fig. 1. Data Flow of the work plan

2 Exploratory Data Analysis

2.1 About Dataset

This file contain behavior data for a one month (December 2019)(2GB) from a large multi-category online store. You can find a link to download the data set in the reference [1].

Each row in the file represents an event. All events are related to products and users. There are different types of events.

2.2 Dataset Structure

Table .1. The Classifiers Comparing

Features	Description
event_time	Time when event happened at (in UTC).
event_type	view-a user viewed a product cart-a user added a product to shopping cart remove from cart-a user removed a product from shopping cart purchase-a user purchased a product Typical funnel: view => cart => purchase
product_id	ID of a product
category_id	Product's category ID
category_code	Product's category taxonomy(code name) if it was possible to make it. Usually present for meaningful categories and skipped for different kinds of accessories.
brand	Down cased string of brand name can be missed
price	Float price of a product present.
user_id	Permanent user ID
user_session	Temporary user's session ID. Same for each user's session. Is changed every time.

3 AdaLoyal Algorithm

A recommendation algorithm which adaptively balances users' purchase frequency statistics and the generalization power from embedding learning methods[2].

Algorithm 1 Pseudo-code of **adaLoyal**

Input: $p_{i,u}, q_{i,u}^{(t)}, C_{i,u}^{(t)}, l_0$.
Output: $\tilde{p}_{i,u}^{(t)}, l_{i,u}^{(t)}$.
for each user u, each item i, each transaction t **do**
  if $q_{i,u}^{(t-1)} = 0$ **then**
    // current item has not been purchased before
    assign $\tilde{p}_{i,u}^{(t)} = p_{i,u}$
    assign $l_{i,u}^{(t)} = l_0$, if $C_{i,u}^{(t)} = 1$; $l_{i,u}^{(t)} = NA$, otherwise.
  else
    // loyalty of current item has been activated
    assign $\tilde{p}_{i,u}^{(t)} = l_{i,u}^{(t-1)} q_{i,u}^{(t-1)} + (1 - l_{i,u}^{(t-1)}) p_{i,u}$
    if $C_{i,u}^{(t)} = 1$ **then**
      assign $l_{i,u}^{(t)} = \frac{l_{i,u}^{(t-1)} q_{i,u}^{(t-1)}}{l_{i,u}^{(t-1)} q_{i,u}^{(t-1)} + (1-l_{i,u}^{(t-1)}) p_{i,u}}$
    else
      assign $l_{i,u}^{(t)} = \frac{l_{i,u}^{(t-1)} (1-q_{i,u}^{(t-1)})}{l_{i,u}^{(t-1)} (1-q_{i,u}^{(t-1)}) + (1-l_{i,u}^{(t-1)})(1-p_{i,u})}$
    end if
  end if
end for

Fig. 2. Pseudo-code of AdaLoyal

3.1 Important Pattern In Users Baskets

- Complementarily between Products.
- Compatibility between user and product.
- User Product Loyalty.

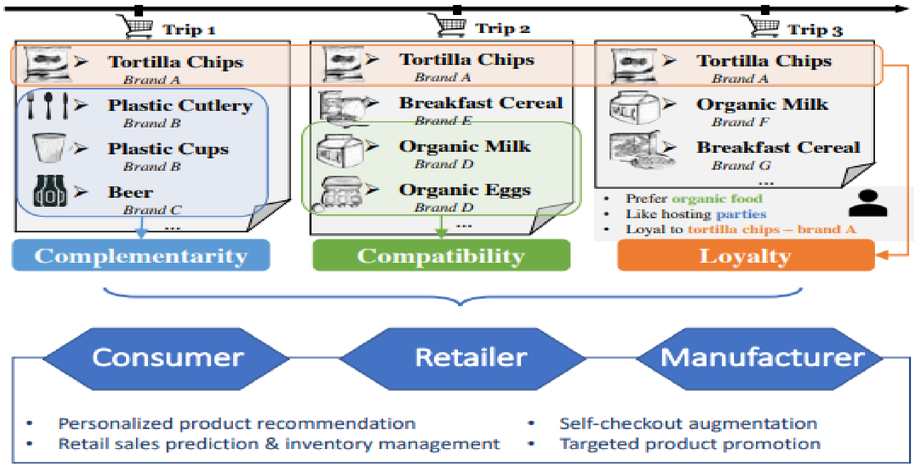

Fig. 3.Three significant patterns observed in users grocery basket

3.2 Learning Models

- item2vec .
- prod2vec.
- metapath2vec .
- triple2vec.

3.3 item2vec

Basket-level skip-grams can be directly applied on this graph, where we treat a particular item as a target node, and the rest of the products in the same basket as contextual nodes. This definition relies on the assumption that products purchased in the same basket share similar semantics, which intuitively supports within-basket/"bundle" product recommendations. However, such co-purchase relationships may not be sufficient to capture personalized preferences toward products [2].

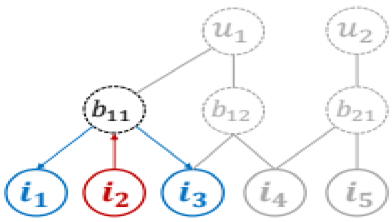

Fig. 4.item2vec

3.4 prod2vec

Rather than directly applying on baskets, in prod2vec,given a target product, the contextual nodes are defined as the product, the contextual nodes are defined as the products in recent baskets purchased by the same user. Unlike the previous method which focuses on within basket co-purchase method which focuses on within basket co purchase relationships regardless of users, this approach emphasizes cross basket item to item relationships for each user [2].

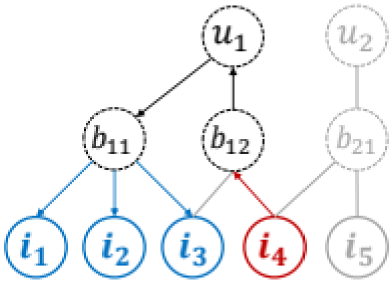

Fig. 5.prod2vec

3.5 metapath2vec

As mentioned, transaction lodes can be transformed in to a heterogeneous network. Therefore, a state of the art network embedding learning method such as metapath2vec can be applied here. In this scenario, we need to defined a symmetric meta path schema:item → basket → user → basket → item,and generate different random walkers based on this predifined scheme.

Specifcally, we start a random product,and sample a series of nodes to compose a random walker where each of the nodes consecutively links to the previous one on this meta path [2].

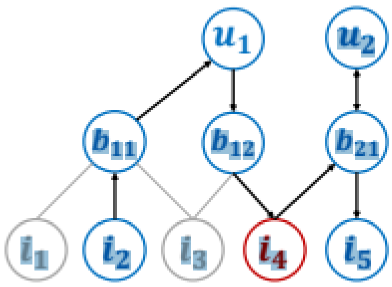

Fig. 6.metapath2vec

3.6 triple2vec

Unlike existing skip gram based product representations, we focus on the cohesion of each (item,item,user) reflecting two items purchased by the same user in the same basket.Specifcally, the transaction logs for training consist of a series of such triples:

$$T=\{(I,j,u) | I \in I_b \wedge j \in I_b \wedge I \;\; j \wedge b \in B_u \wedge u \in U\} .$$

Then we define the cohesion score of each (I,j,u) triple as

$$si,j,u = f_i^T Д_j + f_i^T h_{u + Дj}{}^T h_u$$

where f_i,$Д_j$ are two sets of representations for products and h_u represents the embedding vector of a user [2].

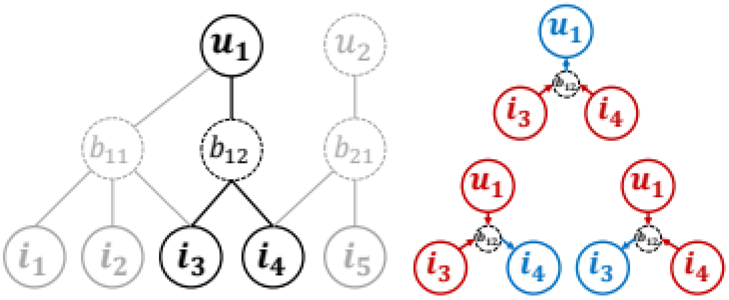

Fig. 7.triple2vec

3.7 Notation

Notation	Description
U, B_u, I_b	user set, basket set for user u, item set of basket b
f_i, g_i, h_u	two different embedding vectors for item i, and the embedding vector for user u
$s_{i,u}$, $s_{i,j,u}$	user u's preference score on item i, the cohesion score of the (item, item, user) triple (i, j, u)
$p_{i,u}$, $p_{i,uj}$	item purchase prob. given a user u, item purchase prob. given a user u and an item j; both are calculated from the original representation learning model
$\tilde{p}^{(t)}_{i,u}$, $\tilde{p}^{(t)}_{i,uj}$	item purchase prob. given a user u for transaction t, item purchase prob. given a user u and an item j for transaction t; both are generated from algorithm 1.
$q^{(t)}_{i,u}$, $l^{(t)}_{i,u}$	user u's empirical purchase frequency and estimated product loyalty of item i up to and including transaction t
$C^{(t)}_{i,u}$	whether product i is purchased by user u in the transaction t

Table 2. Notation Description.

4 Apriori Algorithm

The Apriori algorithm (originally proposed by Agarwal) is one of the most common techniques in Market Basket Analysis. It is used to analyze the frequent item sets in a transactional database, which then is used to generate association rules between the products [3].

C_k: Candidate itemsets of size k
L_k : frequent itemsets of size k
L1 = {frequent items};
for (k = 1; L_k !=ϕ; k++)
    C_{k+1} = GenerateCandidates(L_k)
    for each transaction t in database do
        increment count of candidates in C_{k+1} that are contained in t
    endfor
    L_{k+1} = candidates in C_{k+1} with support ≥min_sup
endfor return U_k L_k;

Fig. 8. Pseudo-code of Apriori Algorithm

4.1 How does the apriori algorithm work

The key concept in the Apriori algorithm is that it assumes all subsets of a frequent item set to be frequent. Similarly, for any infrequent item set, all its supersets must also be infrequent [3].

In order to select the interesting rules out of multiple possible, we will be using the following measures:

- **Support**
- **Confidence**
- **Lift**
- **Conviction**

4.2 Support

Support of the item *x* is nothing but the ratio of the number of transactions in which the item *x* appears to the total number of transactions.

$$\textbf{Support} = \frac{Number\ of\ transactions\ in\ wich\ the\ item\ appears}{Total\ number\ of\ transaction}$$

4.3 Confidence

Confidence (*x* => *y*) signifies the likelihood of the item *y* being purchased when the item *x* is purchased. This method takes into account the popularity of the item *x*.

$$\textbf{conf}(X \rightarrow Y) = \frac{Support(X \cup Y)}{Support(X)}$$

4.4 Lift

Lift (*x* => *y*) is nothing but the 'interestingness' or the likelihood of the item *y* being purchased when the item *x* is sold. Unlike confidence (*x* => *y*), this method takes into account the popularity of the item *y*.

$$\textbf{Lift}(\mathbf{X \rightarrow Y}) = \frac{Support(X \cup Y)}{Support(X) * Support(Y)}$$

4.5 Conviction

Conviction of a rule can be defined as follows:

conv(*x* => *y*) = $\frac{1-supp(y)}{1-conf(x=>y)}$

5 Algorithms Comparison

This table is designed to give you a brief overview in a compare and contrast manner for the two algorithms studied.

Algorithm	Apriori	Ada Loyal
Approach	BFS	Divide and conquer
Scan	Horizontal	Horizontal
Structure	Array	Tree

Table .3. Comparison between two algorithm.

6 Task Description:

- Find products that included in at least 10 different baskets.
- Filter those baskets wish has at least one of this frequent products, take most common products out this basket.
- Take only products that is frequent.
- You take first one out and create train.

7 Solution Idea:

- Group the products id depending on user session.
- Filter the results by using the condition at least 10 different basket.
- Merge the results with orginal dataset to find the most frequent item in each basket.
- Reshape the dataset to a specific format.
- Take out the most frequent product id to create the target values.

8 Splitting data into train and test data

I will split the dataset into training and testing data sets:

- from sklearn.model_selection import train_test_split
- test_size = 0.2
- X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)

The train dataset, which is the largest group, will be used for training, while the test dataset will be used for model evaluation and to test the assumptions of the model.

9 Data Processing

- Group the product id depending on user session

```
df=df_data.groupby(["product_id","user_session"])["product_id"].count()
```

```
df
```

```
product_id  user_session
3762        025d2005-8203-44af-8114-99128476a208    2
            04793e8a-9235-4be1-a0a8-9a4ab0af2298    1
            0d87a4c8-f4bc-47e5-adc0-003ecaecadfb    1
            121d3360-83ea-4898-b6a9-0cbabf39e50c    1
            1fa9816e-5839-4091-ac4d-cc0fdb296550    3
                                                   ..
5909984     c26bf34d-f4a2-4d97-a950-e508b06f1ab8    1
5909985     731bb14d-49eb-488d-bea6-26d8ba5e45c4    1
            f0ef3aeb-ac2b-42fa-9a52-3f2c08d12ba5    1
5909986     802aabe9-a77d-44db-9cf8-d5e335703ca6    1
            88b81b15-0a73-4a9a-b3bc-ac4b3ccb2af9    1
Name: product_id, Length: 112704, dtype: int64
```

Fig. 9. group the product_id

- Filter the results by using the condition at least 10 different basket

```
df['count']=df_data[['product_id']].apply(pd.Series.value_counts)
df['product_id']=df.index.values
df.columns = ['product_id','count']
df=df.loc[df['count'] >=10]
df
```

	product_id	count
5809910	5809910	1435
5809912	5809912	547
5909810	5909810	433
5833330	5833330	422
5700037	5700037	412
...	...	...
5847169	5847169	10
5816540	5816540	10
5561576	5561576	10
5752949	5752949	10
5841719	5841719	10

3739 rows × 2 columns

Fig. 10. filter the product_id.

- Merge the results with orginal dataset to find the most frequent item in each basket.

In [98]:
```
right_merged_total = pd.merge(df_data,df, on=["product_id"],how='right')
right_merged_total.head()
right_merged_total.shape
right_merged_total
```

Out[98]:

	event_time	event_type	product_id	category_id	category_code	brand	price	user_id	user_session	count
0	2019-12-01 00:00:00 UTC	remove_from_cart	5712790	1487580005268456287	NaN	f.o.x	6.27	576802932	51d85cb0-897f-48d2-918b-ad63965c12dc	12
1	2019-12-01 11:29:42 UTC	view	5712790	1487580005268456287	NaN	f.o.x	6.27	449468265	6476495d-b253-4bdf-96a8-ae658346dc89	12
2	2019-12-01 12:12:42 UTC	cart	5712790	1487580005268456287	NaN	f.o.x	6.27	201720695	31054cd3-c9c6-4de2-9560-850bdcd86605	12
3	2019-12-01 12:45:45 UTC	purchase	5712790	1487580005268456287	NaN	f.o.x	6.27	201720695	31054cd3-c9c6-4de2-9560-850bdcd86605	12
4	2019-12-01 13:26:36 UTC	cart	5712790	1487580005268456287	NaN	f.o.x	6.27	580187026	bec28cb9-efbe-4107-96bb-3be577ca5580	12
...	...	...	...	...	...	...	...	...	...	...
98521	2019-12-02 08:08:13 UTC	view	5556128	1487580011853512836	NaN	NaN	8.57	456355035	7e0d6ee4-8c3f-4ff5-bb9c-891e6c7c4caf	10
98522	2019-12-02 08:08:28 UTC	view	5556128	1487580011853512836	NaN	NaN	8.57	456355035	7e0d6ee4-8c3f-4ff5-bb9c-891e6c7c4caf	10
98523	2019-12-02 08:08:37 UTC	view	5556128	1487580011853512836	NaN	NaN	8.57	456355035	7e0d6ee4-8c3f-4ff5-bb9c-891e6c7c4caf	10
98524	2019-12-02 08:08:45 UTC	remove_from_cart	5556128	1487580011853512836	NaN	NaN	8.57	456355035	7e0d6ee4-8c3f-4ff5-bb9c-891e6c7c4caf	10
98525	2019-12-02 08:09:10 UTC	view	5556128	1487580011853512836	NaN	NaN	8.57	456355035	7e0d6ee4-8c3f-4ff5-bb9c-891e6c7c4caf	10

98526 rows × 10 columns

Fig. 11. merge data using right merge.

- Reshape the data frame to the shape that contain 13 columns of product id the take out the most frequent items from them and make the target column.

	user_id	user_session	product_id1	product_id2	product_id3	product_id4	product_id5	product_id6	product_id7	product_id8	product_id9
790	433129158	61	5789608	5751422	5815730	5842141	5887008	5664660	5815659	5865798	5587772
609	377818992	47	5906100	5906118	5877603	5848418	5833330	5833325	5833334	5809323	5809267
619	461173081	48	5804529	5901857	5649213	5804169	5813489	5788520	5873432	5754853	6737
970	580467286	75	5700037	5565816	5700038	5896429	5587658	5587652	5896421	5877490	5862564
1133	548628897	88	5835924	5861278	5304	5742717	60182	5712785	5868798	5732025	5585658
...	...	...	...	...	...	...	...	...	...	...	...
312	440762874	25	5904353	5900284	5833324	5907880	5907851	5907874	5904330	5907839	5907838
933	452834625	72	5809910	5528035	5903137	5739056	5837103	5738964	5738965	5873432	5739036
329	567714739	26	5752069	5807880	5807867	5873620	5847368	5696111	5563596	5852352	5823769
467	415520605	36	5833330	5833326	5877490	5780927	5881777	5839640	5561066	5726236	5901621
726	534827895	56	5692603	5823667	5684434	5677462	5739036	5886536	5769908	5692592	5683950

1332 rows × 17 columns

Fig. 12. Reshape the dataset

product_id10	product_id11	product_id12	product_id13	target	count
5703438	5664642	5751383	5664665	5852144	10
5873627	5809297	5809288	5862313	5846817	10
5730206	5873430	5859402	5528035	5736094	10
5901622	5700049	4591	5758630	5822642	10
5861277	5714118	5343	6817	5560994	10
...	...	...	...	...	...
5843585	5843593	5897978	5908248	5699414	1435
5856946	5739055	5850564	5848388	5877505	1435
5696103	5563536	5861591	5861620	5904751	1435
5875804	5549818	5877491	5857286	5896610	1435
5690214	5852144	5847108	5861597	5889482	1435

Fig. 13. Reshape the dataset

10 Base Line

In order to compare results ,the following models were tried as baseline:

- Random Forest Classifier
- Gaussian NB Classifier
- K Neighbors Classifier

10.1 Random Forest Classifier:

An ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random Forest Classifier was tried with the same data and showed the following result:
Accuracy: **94.93%.**

10.2 K-Nearest Neighbors Classifier:

KNN is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems, it assumes similar things are near to each other, KNN works very fast with the small datasets, aims to finding the distances between a query and all the examples in the data. selecting the specified number examples (K) closest to the query, then votes for the most frequent label.
K-Nearest Neighbors was tried with the same data and showed the following result:
Accuracy: **94.93%.**

10.3 Gaussian NB Classifier:

Gaussian NB are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. They are among the simplest Bayesian network models. But they could be coupled with Kernel density estimation and achieve higher accuracy levels.
Gaussian NB was tried with the same data and showed the following result:
Accuracy: **60.09%.**

11 Decision Tree Classifier:

The decision tree classifier creates the classification model by building a decision tree. Each node in the tree specifies a test on an attribute, each branch descending from that node corresponds to one of the possible values for that attribute.
Decision Tree classifier was tried with the same data and showed the following result: Accuracy: **94.93%.**

12 Classifiers Comparing

Table. 4. The Classifiers Comparing

	Accuracy	**Run Time**
Random Forest Classifier	94.93%	0.12
K-Nearest Neighbors	94.93%	0.2
Gaussian NB Classifier	60.09%	0.3
Decision Tree Classifier	94.93%	0.8

In the following table we compare between the models according to the accuracy and the run time.

13 Matrix Factorization (MF)

Most of the MF models are based on the latent factor model . Matrix Factorization approach is found to be most accurate approach to reduce the problem from high levels of sparsity in RS database, certain studies have used dimensionality reduction techniques. In the model-based technique Latent Semantic Index (LSI) and the dimensionality reduction method Singular Value Decomposition (SVD) are typically combined . SVD and PCA are well-established technique for identifying latent factors in the field of Information Retrieval to deal with CF challenges. These methods have become popular recently by combining good scalability with predictive accuracy. They offers much flexibility for modeling various real-life applications [4].

14 MATRIX FACTORIZATION (MF) MODELS

- Singular Value Decomposition (SVD).
- Principal Component Analysis (PCA).
- Probabilistic Matrix Factorization (PMF).

15 ROLE OF MATRIX FACTORIZATION IN COLLABORATIVE FILTERING ALGORITHM

Collaborative Filtering is a most promising research field in the area of Information Retrieval, so many researchers have contributed to this area. Many CF researchers have recognized the problem of large dataset and sparseness (i.e., many values in the ratings matrix are null since all users do not rate all items), which is been well taken care by Matrix Factorization. Computing distances between users is complicated by the fact that the number of items users have rated in common is not constant. It is important to study the role of Matrix Factorization models like SVD, PCA and PMF with Collaborative Filtering (CF) algorithms. Looking at the contribution of other researchers who have worked in this area have motivated us to work on the role of Matrix Factorization model in Collaborative Filtering algorithm [4].

16 Conclusion

In this paper,The data has been worked on for later use by created the new dataset that contains at least 13 products and the target values column after removed the most frequent products from each column
The following classifiers Random Forests , KNN , Gaussian NB used as a baseline and decision tree used as a base model after that comparing the final accuracy results for each model with another the best accuracy percent was for the Random Forest, KNN and Decision Tree its about **94.93%** and Gaussian NB about **60.09%**

17 References

[1] The instacart online grocery shopping dataset 2019. Accessed on Dec. 2019. https://www.kaggle.com/mkechinov/ecommerce-events-history-in-cosmetics-shop.

[2] MengtingWan, Representing and Recommending Shopping Baskets with Complementarily Compatibility and Loyalty, pp.Jan,2019.

[3] W. Sun, M. Pan, and Y. Qiang, "Improved association rule mining method based on statistical," Application Research of Computers. vol. 28, no. 6, pp. 2073-2076, Jun, 2011.

[4]Yehuda Koren, "Matrix Factorization Techniques for Recommender Sys tems,"Published by the IEEE Computer Society, IEEE 0018-9162/09, pp. 42-49, ©IEEE, August 2009.