Recommendation System for Behance Artistic Community

Xinshu Jiang

Xiangyu Zhuang

Qi Chu

Introduction

Behance is the world's largest social media for people to discover and showcase creative artworks. It has more than 10 million users and there are a wide variety of artworks, including architecture, product design, fashion and so on. Behance users can be both content viewers and content creators. That means, they can not only browse through a large number of various projects and show their compliments by appreciating those they like, but also create and upload their own creative works and let others evaluate them.

In this project, we are trying to utilize various recommendation models to learn about the relationship between users' interaction history and their future actions. The models we covered include unsupervised learning and supervised learning. Specifically, unsupervised learning models are Popularity method, Jaccard-Similarity method, Bayesian Personalized Ranking and Factorized Personalized Markov Chain. Supervised learning models include logistic regression and random forest using features including item popularity, item similarity and items' owner popularity.

1. Exploratory Analysis

OVERVIEW AND METADATA ANALYSIS

Behance is a popular art community where people create, explore and appreciate artworks. The dataset we are using is an anonymized dataset collected from the website by previous researchers [1]. The dataset captures the explicit appreciates sent by users to items with a timestamp and contains information about user-item ownership.

The dataset regarding appreciates of users has a total number of interactions of 1,000,000 and contains three columns. The user-item ownership dataset has a number 18,6560 of pairs. The metadata information can be summarized as follows:

Table 2.1 Appreciate Dataset Overview

Appreciation	Type	Min	Max	Unique Value
user_id	string	-	-	63,497
item_id	string	-	-	178,788
timestamp	timestamp string	2011-06-09 01:34:31	2011-11-14 07:11:14	948,389

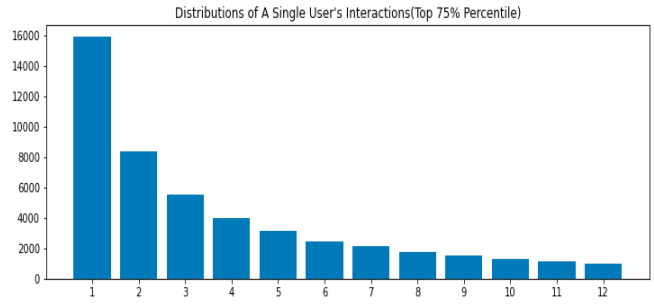
Table 2.2 Item To Owner Dataset Overview

Item_To_Owner_1M.gz	Type	Number of Missing	Unique Value
user_id	string	0	51,487
item_id	string	0	178,788

EXPLORATORY ANALYSIS

Users

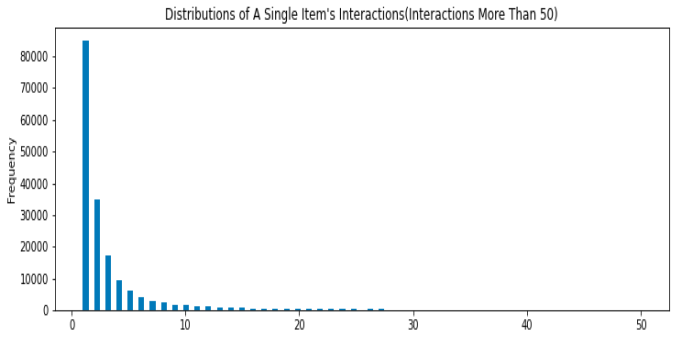
In our dataset, there are a total of 63,497 users. Overall average of interactions of a single user is 15.75, and maximum is 260 and minimum is 1.



Items

In our dataset, there are a total of 178,788 unique items. It's about 2.8 times more than the users, indicating a sparse interaction matrix for our

dataset. The lower quartile is 1, median is 2 and upper quartile is 3. The overall average number of interactions of a single item is 5.59, and the maximum is 1793 and minimum is 1.



Time Session

Among 1,000,000 observations, there are 948,389 unique values of timestamp. Since users' interactions are usually calculated and reported at a certain duration of time in the real world, we decide to take the concept of time session into account and manually divide interactions into a fixed length of time sessions. The table below shows a glance of the statistics of different length time sessions. Ideally, interactions within a session should at least exceed a number of 10. Using this standard gives us two options: time sessions with duration of either 30 minutes or 60 minutes. To ensure abundant observations, we adopt the 60 minutes duration to partition the dataset and obtain a temporal one.

Table 2.3 Statistics of Different Length Time Session

Session Duration	Max Interactions	Min Interactions	Number of Sessions
5	86	1	45,550
10	148	1	22,781
15	205	2	15,187
20	270	7	11,391
30	394	18	7,594
60	**749**	**51**	**3,798**

Ownership

Behance allows works of collaboration on its website, which indicates a possibility of multiple workers of an item. In the dataset, there are 178,788 unique items and 51.487 unique users. Among all items, about 3.33% (5954) of them are workers created by multiple users. Among all creators, the majority of them are less than 5, with the upper quartile being 4. The average number of artworks created by a single user is 3.62 in this dataset. Among all users (users in appreciates dataset or ownership dataset), about 0.7% are both content creators and appreciators, content appreciators take up to 55.61% of the total users, which is slightly more than the percentage of content creators (45.09% with 50,684 users).

Table 2.4 User Matrix

User Matrix	Content Appreciator |Yes	Content Appreciator | No	Total
Content Creator | Yes	803 (0.7%)	50,684 (44.39%)	51,487 (45.09%)
Content Creator | No	62,694 (54.90)	0 (0%)	62,694 (54.91%)
Total	63,497 (55.61%)	50,684 (44.39%)	114,181 (100%)

2. Predictive Task

Based on this selected dataset, our goal is to predict which item a certain user is most likely to appreciate given previous action history. Our dataset contains over 1Million interaction records. The nature of our predictive task requires that our models rank the ground-truth item higher than other items or negative examples for a specific user. We choose AUC score as the evaluation metric of our models and to validate outputs from our models. Our AUC function randomly chooses a positive instance from the dataset and generates a negative instance at the same time, then see if the classifier ranks the positive case higher than the negative case. For this predictive task, there are several models we believe are appropriate to use. In terms of unsupervised learning models, we use models based on Jaccard similarity, Bayesian Personalized Ranking and MarkovChain. For supervised Learning, we explore LogisticRegression and Random Forest. We will discuss model-specific features and input in the later Model Information and Design section.

3. Related Work

In this section, we will briefly discuss related research on existing models that are applicable to our predictive task.

Jaccard Similarity

In related research, various traditional similarity approaches like cosine similarity, Pearson's correlation, Jaccard similarity and several data mining techniques have been discussed a lot to classify nearest neighbors and generate recommendations. Although most of the techniques are complex or time-consuming and further improvements have been observed, the purpose of the similarity model is to identify appropriate nearest neighbors of the targeted user and predict accurate ratings of unrated items.

$$Sim(u,v)^{Jaccard} = \frac{(I_u \cap I_v)}{(I_u \cup I_v)}$$

It is the ratio of the proportion of the cardinality of co-interacted items to the cardinality of all items interacted by both the user.

Bayesian Personalized Ranking (Implicit)

The Bayesian Personalized Ranking is a pairwise method and has already been widely used in the recommender systems world to realize personalized ranking. BPR derive from the MF[9], that decomposes the user-item rating matrix R into user embedding matrix W and item embedding matrix H. Based on implicit feedback, it is assumed that the observed interactions should have a ranking that is higher than the unobserved ones. This is achieved by training a predictor xu,i, j that assigns a score based on which of the two items. (i or j) is preferred.

Markov Chain

In most of the recommendation system models, we rely on the historical data of user-item pairs to learn and model a pattern and therefore try to predict or recommend the next item that the user will interact with. Moreover, in some cases, the best predictor of the next item that user will choose to interact with is very likely to be the most recent item or a list of items in the most recent time sessions. Markov Chain is exactly the model that could capture the contextual information and personalized information under improvement. Factorized Markov Chain (FMC) is a Markov chain model improved with the technique of matrix factorization to summarize the structure of the matrix and discover the implicit features of items instead of relying on more features. The idea of matrix factorization can be denoted as $R_{i,j} = \gamma_i \cdot \gamma_j$. Factorized Personalized Markov Chain (FPMC) is similar to FMC but takes account of user information in the model.

In the past, Markov Chain models have been proved to be successful at capturing both sequential and personalized information and therefore an ideal model for recommendation systems. Rendle et al. [6] brought up the idea of Factorized Personalized Markov Chain (FPMC) and proved that their model outperformed models using just matrix factorization technique and basic markov chain models. To deal with classic problems of sparse, long-tailed datasets, and cold-start problemsCai et al. [7] improved the Factorized Personalized Markov Chain and proposed a new method to incorporate social information to improve the performance. Zhang et al. [8] also adopted Markov Chain to build a personalized recommender system capturing both short-term and long-term preferences to make sequential recommendations of the next song.

Logistic regression

So far, logistic regression has been widely used in recommendation systems. [1] and etc. from Yahoo! applied the logistic regression model and collaborative filtering to build an advertising recommendation system. [2] and etc. from Stanford University implemented a news recommendation system using logistic regression and Naive Bayes classifiers.

Random forest

[10] and etc. from Rakuten Institute of Technology used random forest model for a content-based recommendation systems for E-Commerce Offers and Coupons in order to handle noisy features and proved that random forest are robust to outliers and are capable of handling missing values.

4. Model Information and Design

Train/Val/Test Split

Firstly, we filter the dataset and select the interaction data of those users with at least three interactions since we need at least 3 observations to split into train, validation and test set. After filtering, we construct a validation set and a test set.

Time Session Partitioning

As mentioned in the exploratory analysis, we decided to incorporate the concept of time session and mimic it by manually partitioning our dataset using a duration of 60 minutes. In this case, we have a total of 3,798 sessions, with the minimum interaction of 51 and maximum interaction 749 in one session.

Model Information

Our models and their features can be quickly summarized as below:

Table 5.1 Model Comparison

Model	Personalized?	Temporally-aware?	Ownership-aware?	Supervised-Learning?
S-POP	N	Y	N	N
Jacc	N	N	N	N
BPR	Y	N	N	N
BPMC	Y	Y	N	N
LR	Y	N	Y	Y
RFC	Y	N	Y	Y

Session-based Popularity(S-POP)

It's very intuitive for us to start building our recommender system with a trivial predictor, which also forms a baseline for more complex models. Taking temporal information, we build a recommender system by just recommending the most popular items in the time session. In this case, we define the session-based popularity of an item as the total number of appreciates in the given session.

Let i denotes the given item and t denotes the given session, session-based popularity can be denoted as:

$$Popularity_{i,t} = Number\ of\ Total\ Clicks$$

Jaccard Similarity (Jacc)

The Jaccard Similarity basically computes the similarity between 2 sets. In this specific setting, the goal of the model is to find the maximum item to item Jaccard Similarity given a pair of user and item. We first build corresponding data structures to store the sets of items appreciated by each user and also the sets of users who have appreciated each item -- Iu and Ui.

Given an item i and item j, we implement the Jaccard similarity straightforwardly through the below formula:

$$Jaccard(i, j) = \frac{\left|U_i \cap U_j\right|}{\left|U_i \cup U_j\right|}$$

For the target user, we calculated the Jaccard similarity between the target item and every item the user has previously appreciated then got the maximum similarity score. This is based on the assumption that positive instance (real user-item interactions) is supposed to have a higher maximum item-to-item similarity compared with the negative instance. We achieved the AUC of 0.638 on the validation set, the performance is less satisfactory compared with models we will discuss later.

Bayesian Personalized Ranking

For the implicit Bayesian personalized ranking model, we get some insights from the sample code from chapter 5 workbook. Since we'll be converting the data to a sparse interaction matrix (a matrix in which most of the elements are zero.), the main data structure here is to assign each user/item to an ID from 0 to nUsers/nItems through the usage of dictionary (userID/itemID as key and index as values). We then mark existing interactions between certain users and items as 1. We utilized the bpr.BayesianPersonalizedRanking and fit the matrix.

itemFactors = model.item_factors
userFactors = model.user_factors

By executing the above 2 commands, we get the index associated with each individual item and user

within our training set. We could then use matrix multiplication to get the ranking for an item given a user:

np.matmul(userFactors[userIDs[u]],itemFactors[itemIDs[i]])

For every user in our testset, we will be able to get the u. For items that don't exist in the train dataset but appear in the validation set, we failed to locate the i . at first. We then utilized the mean ranking of all available items for the target user in the face of a cold start. For this model, we tuned the number of factors, model learning rate and regularization. We achieved the AUC of 0.7635 on the validation set with 7 factors, learning rate of 0.1 and regularizer of 0.001.

Factorized Personalized Markov Chain

For the Markov Chain model, we decide to adopt the Factorized Personalized Markov Chain (FPMC) since user characteristics of artists often prove to be essential and user information is abundant in the dataset.
The model relies on the assumption that the previous appreciation by the user will be of great help to predict the next one. Let i represents the next interaction item, j represents the previous appreciated item and u represents a specific user. The assumption can be described using a function denoted as:

f(i | u, j)

Integrating the matrix factorization technique to present the compatibility of user u and item i and the compatibility of user u and item j, the function can be further described as:

f(i | u, j) = i+u(ui)i(iu)+i(ij)j(ji)

The model is then optimized using a BPR-like framework, i.e. using a contrastive loss of the form

(f(i | u, j) - f(i' | u, j)),

where i'is a sampled negative item that the user did not consume.

For cold-start items, we will use a naive way to initialize the item with 0.

Supervised Learning

For training supervised learning models, we need to generate negative samples and construct relevant features.

Negative sample generations:

For each (user, item) interaction in the training set, we randomly sample one item among all the items in the training set. If the user has appreciated this sampled item before, then we re-sample until we obtain one item that the user has not appreciated. In this way, we have all the positive and negative samples for the training set.

Relevant features construction:
this way, we have all the positive and negative samples for the training set.

Table 5.2 Relevant features

Features	Definition	Calculation base
Item_popularity	Popularity of each item	Number of appreciations for each item
max_Jaccard_similarity	Maximum Jaccard similarities between the item and all other items the user has appreciated	Definition of Jaccard similarity, Users set who have appreciated each item
owner_popularity	Popularity of each owner	User-Owner relationship,number of appreciations for each owner

Detailed calculation:
item_popularity:
Since we have our training dataset of interactions, we can count the number of appreciations for each item. Then we can sort and rank the items according to the number of appreciations in reverse order, which gives us the list of items in descending order of corresponding number of appreciations. Next we can calculate item popularity of each item by

$$\frac{\sum_1^j count_{item_i}}{\sum_1^n count_{item_i}}$$

, where j represents the item's rank according to the number of appreciations in reverse order, and n represents the total number of items. If the item does not appear in the training dataset, we set its item popularity to be 1.

max_Jaccard_similarity:
In order to calculate Jaccard similarities between any two items, firstly we need to create a dictionary to store the information of all users who had appreciated it for each item. Then we can just

calculate the Jaccard similarity according to the user sets for each item.

owner_popularity:
During the calculation of the first feature, we have got the number of appreciations for each item. Since we also have the item-owner relationship from the 'Item_to_Owners' Dataset, we can use that to obtain the number of appreciations for each owner. As for how we calculate item popularity, similarly, we can sort and rank the owners according to the number of appreciations in reverse order. Then we can calculate owner popularity of each owner by

$$\frac{\sum_1^j count_{owner_i}}{\sum_1^{n_{owner}} count_{owner_i}}$$

, where j represents the owner's rank according to the number of appreciations in reverse order, and n_{owner} represents the total number of owners. If the owner does not appear in the training dataset, we set its owner popularity to be 1. If there are more than one owner for the item, owner popularity for that item is the average of owner popularity for each owner.

Using the calculations mentioned above, we can compute these features for positive and negative samples in the training dataset. Then we create a label called 'appreciated' as our response variable. For all positive samples, 'appreciated' = 1. While for all negative samples, 'appreciated' = 0.

Model training & testing:
Now that we have our training X as 'item_popularity', 'max_Jaccard_similarity', and 'owner_popularity' features from the training dataset, and training Y as 'appreciated' feature. We can use supervised learning models, namely logistic regression and random forest, to fit the training data to train our model.

In order to calculate AUC score in the test set for each user, we need to know whether the classifier will rank the positive samples higher than the negative samples or not. Therefore, we also need to calculate the 3 features above in order to use our trained model. Here we use predict_proba() function for logistic regression and random forest classifier using sklearn package.
Since predict_proba() gives us the probability estimates for each observation to be class 0 or 1(response variable = 0 or 1), we can use it to compare the probability estimate of class 1 for positive and negative samples and see if the probability estimates for positive samples are higher than that or negative samples or not.

Logistic Regression (LR)

Logistic regression is a basic statistical model that utilizes logistic function to model a binary response variable. After fitting the model on the training set, based on the model performance on the validation set, we tune model parameters including penalty and C(regularization strength).

Random Forest Classifier (RFC)

Random forest model utilizes ensembling technique to combine the result of a large number of decision trees. Similarly, utilizing the model performance of trained model on validation set, we tune the model parameters including 'n_estimators'(number of trees), 'criterion'(function to measure quality of a split) and max_depth(maximum depth of the tree)

Evaluation Metric

As mentioned earlier, our validation and test set contain the last 2 interactions of users who have at least 3 interaction records. All 6 methods were trained on the same training set with parameters tuned on the validation set. Our model performance is based on the trained models' corresponding performance on the test set. Since our objective is to predict which item the user is most likely to appreciate given past interaction history, a reasonable evaluation would be to see if the ground-truth item has been ranked higher than randomly generated negative case. For all of our models, we choose AUC score as the evaluation metric to follow a consistent comparison scheme. For each user in the testset, our AUC function randomly chooses a positive instance i and negative instance j at the same time, then see if the classifier ranks the positive case higher than the negative case.
For a specific user u we have:

$$\mathrm{AUC}(u) = \frac{1}{|I_u||I \setminus I_u|} \underbrace{\sum_{i \in I_u}}_{\text{positive items for user } u} \underbrace{\sum_{j \in I \setminus I_u}}_{\text{unseen items for user } u} \delta(x_{u,i,j} > 0).$$

For an entire dataset we average the above across all users:

$$\mathrm{AUC} = \frac{1}{|U|}\mathrm{AUC}(u)$$

Based on the above calculation, the AUC score of a given model is supposed to be between 0 and 1, where an AUC of 1 means that the model always

ranks positive items higher than unobserved items; an AUC of 0.5 indicates that the model is no better than recommending items randomly. AUC is thought to be a better measure than simple accuracy as it is based on a balanced test dataset and takes the ranking into consideration.

5. Model Performance and Analysis

As mentioned before, popularity and Jaccard takes into account least aspects when predicting the appreciated item of users. Therefore, they are the baseline of the rest complex models. From the table, it's clear that all of the complex models beat the baseline models' AUC by at least 0.1.

Supervised Learning vs. Unsupervised Learning

The traditional recommender system models adopted by us are the unsupervised learning models such as Bayesian Personalized Ranking(BPR) and Factorized Personalized Markov Chain(FPMC). Besides that, we also include some supervised learning models with heuristics like popularity and Jaccard similarity. In the table, we found un-supervised learning models have a balanced performance compared to supervised learning models. BPR's performance is similar to LR and it slightly outperforms LR by 0.0012. FPMC's performance is similar to RFC and it outperforms RFC by 0.03.

Based on research work, we expect that FPMC, being a personalized, factorized and temporally-aware, would have the optimal performance on validation dataset. However, it turns out that the Random Forest Classifier is the optimal model with the AUC performance of RFC. We speculate that there are two reasons:

First, FPMC could be improved by updating the initialized value of item and parameters. Instead, we just use a naive value of 0.

Second, the users of the Behance dataset are artists. Artworks and artists tend to show a strong personal preference. Behance dataset is very likely to be an ownership-dominant dataset. Therefore, RFC, taking the ownership information, is of great chance to capture the characteristics of the specific Behance dataset. Future works of FPMC on the dataset could consider adding variables representing the owner-item and owner-user compatibility.

Table 5.3 Model Performance

Metric	S-POP	Jacc	BPR	FPMC	LR	RFC
AUC	0.512	0.638	0.763	0.858	0.762	**0.888**

6. Conclusion

By adopting the dataset obtained from Behance dataset, we build the recommendation systems using both unsupervised learning and supervised-learning models that take into account the personalized, temporally-aware and ownership-aware features. We split the dataset into train, test and validation set. We train and find the best hyper-parameters using the train and test dataset. The performances of our models are reported on the validation dataset. The optimal model in our case is the Random Forest Classifier(RFC) with AUC of 0.8888 that captures both personalized and ownership-aware information of the data. We also discuss the reasons why the estimated optimal model of Factorized Performance Markov Chain(FPMC) doesn't outperform the RFC. Based on the discussion, we point out the directions of future improvement of our models.

Reference

[1] Bartz K, Murthi V, Sebastian S. Logistic regression and collaborative filtering for sponsored search team recommendation[C]//Second workshop on sponsored search auctions, 2006, 5

[2] Lau C W. News Recommendation System Using Logistic Regression and Naive Bayes Classifiers [J], 2011.

[3] Vista: A visually, socially, and temporally-aware model for artistic recommendation

[4] Ruining He, Chen Fang, Zhaowen Wang, Julian McAuley

[5] RecSys, 2016

[6] Rendle, S., Freudenthaler, C., & Schmidt-Thieme, L. (2010, April). Factorizing personalized markov chains for next-basket recommendation. In Proceedings of the 19th international conference on World wide web (pp. 811-820).

[7] Cai, C., He, R., & McAuley, J. (2017). SPMC: socially-aware personalized markov chains for sparse sequential recommendation. arXiv preprint arXiv:1708.04497.

[8] Zhang, K., Zhang, Z., Bian, K., Xu, J., & Gao, J. (2017, June). A personalized next-song recommendation system using community detection and markov model. In 2017 IEEE

Second International Conference on Data Science in Cyberspace (DSC) (pp. 118-123). IEEE.

[9] J. Wang and P. Han, "Adversarial Training-Based Mean Bayesian Personalized Ranking for Recommender System," in IEEE Access, vol. 8, pp. 7958-7968, 2020, doi: 10.1109/ACCESS.2019.2963316.

[10] Xia, Y., Di Fabbrizio, G., Vaibhav, S., & Datta, A. (2017). A Content-based Recommender System for E-commerce O ers and Coupons. In Proc. SIGIR Workshop eCommerce