Assignment 2: Addressing Cold Start Problems with Content-Based Filtering for Steam Dataset

Miao Hao, Jianghong Wan

I. Introduction

"Cold start" problem is one of the potential challenges in recommender systems. It refers to the cases where the model performs poorly because the user or item in inference is not seen in training. In this assignment, we have designed a recommender system specializing in cold start problems. Specifically, we focus the item cold start problems, i.e. the recommender needs to predict for unseen items. We have picked the *Steam* [1]–[3] dataset. In this case, the item cold start problem occurs at scenarios where new games are released. We want to design a model that can recommend these new, unseen games to the users.

In Assignment 1, we have known that the Collaborative Filtering Model is very strong in this domain. However, this model will be greatly affected by cold start problems. This is because most information the model acquires from the training set relies totally on each specific user and item. For item cold start cases, the information left for inference is very limited.

To address this problem, we utilize the additional information of the new items. For example, on Steam games scenario, when a new game is released, its genre (e.g. Action, Racing, RPG) will also be released by the developer. By replacing the items with their corresponding genre in the model, we are able to bypass the cold start problem, since we may assume that there will be no new genre. This method is also known as Content-Based Filtering.

To demonstrate the effectiveness of our method, we have built a dataset specifically for item cold start problems from Steam dataset. We also build a linear model as baseline. By evaluating on our built dataset, we show that our model is able to surpass the baseline and is more effective.

II. Dataset Analysis

In this assignment, we are using **Steam Video Game** dataset [1]–[3] (https://cseweb.ucsd.edu/~jmcauley/datasets.html#steam_data). There are five datasets of steam platform, two of which drew our interests: Australian user-item data, which contains about 88,000 sets of information regarding users from Australia, such as how many games they have played, and basic info about each of those games, etc.

Another dataset, metadata, contains more detailed information for each of the 30,000 games, such as name of the publisher, genres it belongs to, and pricing, etc.

For the Australian user-item data, there are 71,500 users who have a steam id and has played at least one game. The average number of games of those users are around 72. The player with the greatest number of games played is "phrostb"


```
{'user_id': 'jorellpogi',
 'items_count': 71,
 'steam_id': '76561198103696236',
 'user_url': 'http://steamcommunity.com/id/jorellpogi',
 'items': [{'item_id': '240',
   'item_name': 'Counter-Strike: Source',
   'playtime_forever': 3519,
   'playtime_2weeks': 0},
  {'item_id': '4000',
   'item_name': "Garry's Mod",
   'playtime_forever': 28218,
   'playtime_2weeks': 95},
```


Fig. 1: User-game data example


```
{'publisher': 'Kotoshiro',
 'genres': ['Action', 'Casual', 'Indie', 'Simulation', 'Strategy'],
 'app_name': 'Lost Summoner Kitty',
 'title': 'Lost Summoner Kitty',
 'url': 'http://store.steampowered.com/app/761140/Lost_Summoner_Kitty/',
 'release_date': '2018-01-04',
 'tags': ['Strategy', 'Action', 'Indie', 'Casual', 'Simulation'],
 'discount_price': 4.49,
 'reviews_url': 'http://steamcommunity.com/app/761140/reviews/?browsefilter=mostrecent&p=1',
 'specs': ['Single-player'],
 'price': 4.99,
 'early_access': False,
 'id': '761140',
 'developer': 'Kotoshiro'}
```


Fig. 2: Metadata example

with 7762 games played. The most time a user spent on a game is by 'wolop', and the game is 'Garry's Mod' with 642773 unit of time. The unit of time is not specified, but we made a reasonable guess that it is in seconds, which leads to about 178.5 hours. The user who spends the greatest amount of time on all games is "REBAS_AS_FT", who spent 4660393 seconds, about 1294.5 hours in total on games.

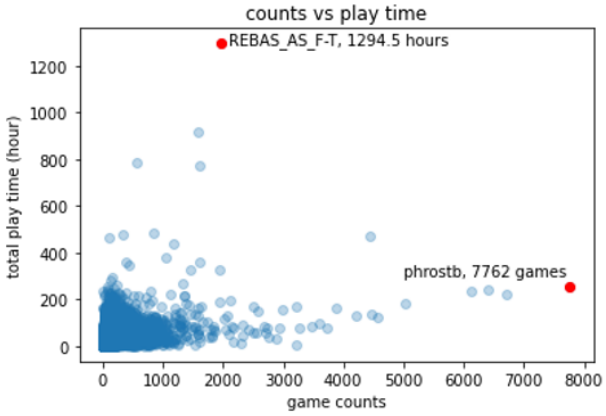

Fig. 3: Total play time vs game counts

To study the connections between these two datasets, we have to filter the datapoints. There are over 30,000 games in

the metadata dataset, and we set the following conditions for filtering:

- It has a game id
- It presents in at least one of the Australian users' play list
- It has at least one genre

We discard all datapoints that don't satisfy the above conditions. Among over 30,000 games in metadata, there are about 8600 games that meets the requirement. Interestingly, the top 3 most frequent genres for games are: Indie, Action, and Adventure, they appeared in 5327, 3863, 3003 games respectively, out of those 8600 games.

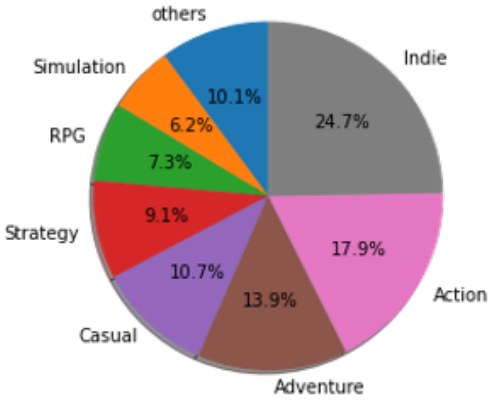

Fig. 4: Pie chart for genre distribution

The top 3 most common combination of genres are: "Action, Indie", "Action", and "Action, Indie, Adventure", which is not surprising considering our previous finding. An interesting finding is that there are 23 genres in total shared by over 20,000 games. By combining these two datasets, we can easily map users to genres and publishers.

Addtionally, there is a very interesting study on the genres, tags, and their networks [4], where the authors suggested that video game genres are more complicated than simple classification, and should be further studied via network analysis.

III. Predictive Task

As we have stated in Section I, the goal of our assignment is to propose a model specializing in addressing the item cold start problems. We select our predictive task, evaluation metric, and build our dataset all based on this goal.

Given the characteristics of the *Steam* [1]–[3] dataset, we want to build a recommender system that can predict whether a given user would purchase a specific game. In other words, we want the model to generate a function $p(u, i)$, where u is the given user, i is the given game, or item, and $p(u, i)$ is the probability that u will purchase i. With this model, we may recommend the games to Steam users that they are likely to purchase.

In order to focus on the item cold start problem, the dataset we use needs special modification. Specifically, we let the training set and the testing set have no common items. In other words, for the item set in training I_{train} and item set for testing I_{test}, we have $I_{train} \cap I_{test} =$. Under this setting, all the items the model needs to inference are unseen in the training stage, fully validating the model's ability in item cold start circumstances. Also, according to the analysis on the dataset in Section II, the whole Steam dataset is quite large. Due to the restraint of our computational resource, it would be difficult for us to conduct experiments if we utilize the whole data. So we decide to use a subset of the dataset. In specific, we pick the first 10,000 users and the first 10,000 games, and filter out the rest. Because the each data pair needs to satisfy both, the final number of users and number of games is smaller. After the filtering, there are 7193 users, 2690 games, and 153814 user-game pairs. We use the above subset as the training set. We then use the same users, and pick another 1,000 games (i.e. 10,000 - 11,000). We use this subset as the testing set. Testing set has 272 games and 15875 user-game pairs.

As we have breifly introduced in Section I, the key of our method is to utilize additional information of the unseen items. In our assignment, we decide to use the genre information of each game. For each data pair (u, i), where u is the user and i is the game, we transform it into (u, G_i), where G_i is the genre set of game i (a game typically has multiple genres). We use this data format in both our model and the baseline model that we will introduce later.

We use multiple evaluation metrics to validate the models. Firstly, we use the same evaluation metric from assignment1. Since we have only positive data pairs in the dataset, we need to generate negative data pairs as well. For each data pair (u, i) in the testing set, we randomly pick a game i', which u has not purchased, and add (u, i') to the testing set as the negative sample. After this, we predict our model on all the samples in the testing set, and compute its accuracy. In the rest of this report, we denote this metric as "Accuracy".

We also use mAP@k (mean average precision) and mAR@k (mean average recall) as the evaluation metrics in our assignment, which are common metrics for recommender systems. For each user in the testing set, we recommend k items to them (obviously, we need to pick the k items that have the highest probability this user would buy). Denote this set of size k as $\hat{I}_u^k$, and the game set that this user actually purchased (the ground-truth) as I_u, we have

$$AP@k(u) = \frac{|\hat{I}_u^k \cap I_u|}{k} \tag{1}$$

and

$$AR@k(u) = \frac{|\hat{I}_u^k \cap I_u|}{|I_u|}. \tag{2}$$

Calculating the mean on the whole testing set, we have mAP@k and mAR@k, i.e.

$$\begin{aligned} mAP@k &= \sum_{u \in U} AP@k(u) \\ mAR@k &= \sum_{u \in U} AR@k(u), \end{aligned} \tag{3}$$

where U is the user set in testing. In this assignment, we use $k = 10$.

To better analyse the effectiveness of our method, we use a linear model as baseline. We also use the genre set as the

input. We represent each genre set as a vector, and we fit the model using `LogisticRegression` model in sklearn.

IV. Method

From assignment 1, we have known that a Collaborative Filtering Model as follows is very strong:

$$p_{u,i} = \mu + \beta_u + \beta_i + q_i^T(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j), \quad (4)$$

where u is the user, i is the game, $N(u)$ is the games that u has played, and $p_{u,i}$ is the predicted probability that u would purchase i.

However, on item cold start scenarios, it is easy to find that this model will perform poorly. The last term of Eq. 4, known as user implicit feedback, is not available, because information on the item i cannot be acquired from training since i is an unseen new item. It is similar for the term β_i as well. Without these two terms, the model becomes

$$p_{u,i} = \mu + \beta_u. \quad (5)$$

It is obvious that this model would have little predicting ability.

Our solution is to utilize the additional information that can be acquired even though the item is new, for example *genres* of the game in Steam case. It is known that, when a new game is released, its genres, e.g. Action and Indie, will also be released. We can also assume that all of these genres are known. So there will be no cold start problem for genres. Thus, we aim to replace the item terms in Eq. 4 with the corresponding genre set.

In Eq. 4, there are three terms with regard to the items, β_i, q_i and y_j. For each item in the dataset, there is this scalar and two vectors. We now need to transform them into vectors with regard to the genres, instead of items. A simple solution would be linearly combining the factors of genres:

$$\beta_i = \frac{1}{|G_i|} \sum_{k \in G_i} \beta'_k, \quad (6)$$

where G_i is the genres of item i. Similarly, we may transform q_i and y_j as follows:

$$q_i = \frac{1}{|G_i|} \sum_{k \in G_i} q'_k, \quad (7)$$

$$y_j = \frac{1}{|G_j|} \sum_{k \in G_j} y'_k. \quad (8)$$

In this way, we may rewrite Eq. 4 into

$$\begin{aligned} p_{u,i} = \mu + \beta_u + \frac{1}{|G_i|} \sum_{k \in G_i} \beta'_k + \\ (\frac{1}{|G_i|} \sum_{k \in G_i} q'_k)^T (p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} \frac{1}{|G_j|} \sum_{k \in G_j} y'_k). \end{aligned} \quad (9)$$

The above equation summarizes our method. In this transformed equation, no term relies directly on the item. They rely on the genres instead. Thus, our model can predict new, unseen items with no difficulty. In fact, since the number of genres are much less than the items, we presume it would be more robust. Note that the transformed equation is still differential, allowing us to optimize our model with gradient descent.

We now describe the other details of our method. For the size of the latent vectors in the model, we use 16. In the optimization process, we use Adam optimizer, as it appears to be more stable than SGD. We use cross entropy loss as the objective, which is a common loss function for binary classification tasks. We use batch size of 1024. The learning rate is 10^{-3}. We use weight decay (L2 regularizer) of 5×10^{-5}, otherwise the model overfits. For each positive user-game pair in training set, we pick another game that the user has not played as a negative sample (the same method of our first evaluation metric). We train our model on the training set for 50 epochs.

V. Related Literature

There are three literatures that used the related dataset as ours in the past. The first one [2] by Pathak, Gupta, and Prof. McAuley, aimed to recommend existing steam game bundles and generate personalized bundles, which is robust to the "cold bundles" [2]. The study focuses on the "bundle" aspect of the dataset, which contains the bundle features such as bundle size and compatibility. The method used for existing bundle recommendation was Bayesian Personalized Ranking[5]. The format of the training data was a triplet of (user,positive item, negative item), where item could be a single game when training the item BPR, or a bundle when training bundle BPR. Similar to our approach, where the cold games can be recommended based on their genres, this model can handle "cold item" problem, since the models are trained based on the features of the bundles other than bundle itself. Also, a greedy algorithm was proposed to generate personalized bundles, and the authors came up with a bundle generation model that qualitatively meets users' preferences, with bundle size and bundle correlations as features. An interesting point that this study concluded agrees with our project: By transforming the user-item recommendation to user-features recommendation, the cold start item problem would be solved with little sacrifice on performance.

Another study on this dataset was to study the inner relationship between implicit and explicit signals [1]. It focuses on the Australian user dataset, and exploring based on features like game purchased,play time, reviews, thump-up or thump-down to build a "purchase – play – review – recommend" chain.[1] It shows quantitatively and qualitatively that many users' interactions follow a monotonic structure, such that a more explicit feedback from users often implies that there is an implicit signal present. A study of sequential recommendation also used this dataset [3], while they discarding users who reviewed a game less than 5 times or items that has been reviewed less than 5 times. A review is modeled as an user implicit feedback, and then those cations are sequentially ordered by time. The list of actions are partitioned as: most recent, second recent, and the others. The authors proposed a new approach to sequential recommendation, SASRec, which is a self-attentive sequential recommendation that balance the long-term and short-term semantics.

There are many researches on similar databases as well, for example, Netflix movie recommendations. In fact, the model we used was very closely related to the SVD++ model proposed by Koren *et al.* [6], a model implementing Matrix Factorization (MF) techniques [7] . This model evolved from a simple SVD model with biases, with addition of a user implicit feedback term. As in the report that Yzu-Wei Sung shared on Piazza [8] regarding the Netflix competition winning algorithm, Koren *et al.* further investigated the model and accounted for temporal features of the dataset: the dynamic of movie popularity, change of user baseline and preferences ratings over time.

$$\hat{r}_{ui}(t) = \mu + b_u(t) + b_i(t) + q_i^T(P_u(t) + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j) \tag{10}$$

where they studied the time variant characteristic of user/item biases and user implicit feedback, based on Netflix movie dataset.

Other than Latent Factor models, they also implemented several collaborative filter methods (neighborhood models [9]). Starting with a simple model which predict ratings based on the weighted average of ratings in the neighborhood items, they first improved the weights to be interpolation weights Eq. 11, where wij is the global weight term.

$$\hat{r}_{ui} = b_{ui} + \sum_{j \in R(u)} (r_{uj} - b_{uj}) w_{ij} \tag{11}$$

A further improvement was made by removing the user-specific weights and using the global weights which is user-independent. Their final CF model suggested is shown in Eq. 12, which is a neighborhood model with global weights and regularized implicit feedback.

$$\begin{aligned} \hat{r}_{ui} = \mu + b_u + b_i + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) w_{ij} \\ + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} c_{ij} \end{aligned} \tag{12}$$

where $R(u)$ is the set of ratings by users.

The model we used is very similar to the SVD++, but due to the property of the dataset, they studied the problem of recommendation based on user-item pairs, other than user-item feature pairs, which leads to less effective predictions toward cold items.

Many other researches on cold start problem has been conducted, using predictive content filtering [10], collaborative filtering [11], and functional matrix factorization [12].

VI. Results

In this section, we demonstrate the effectiveness of our method with experiments on our built dataset.

In Table I, we show the general results on our model and baseline model, as well as an ablation on implicit feedback. As we stated in Section III, we use several metrics to evaluate our model. On all three metrics, our model shows significant improvement over baseline model, because the baseline model (a linear model using genres) does not consider user. The second row shows the result of our model without implicit feedback (by removing the last term of Eq. 9). Because it does not consider the relations between user and genres, it also performs poorly compared with our method. We also discover that mAP@k and mAR@k are much more sensitive than "Accuracy", the metric we used in assignment1. Although our model is slightly better than other alternatives on "Accuracy", our model shows massive improvement on mAP@10 and mAR@10.

Model	Accuracy	mAP@10	mAR@10
Linear Model	0.6757	0.0158	0.0101
Ours (w/o implicit feedback)	0.6854	0.0173	0.0715
Ours	**0.6877**	**0.0545**	**0.1948**

TABLE I: Results of several models we evaluated in this assignment. The best results are shown in bold face.

It is also worth noting that, one may think the mAP and mAR we show in Table I is very low. This is because we only use a subset of *Steam* dataset due to the restraint in our computational resource. There are only 272 games in our testing set, roughly 1/30 out of the total game numbers, making the available recommending options much smaller than in real situation. However, even if the exact value of mAP and mAR seems low, we believe it can still show the big advantage of our method over the alternative models.

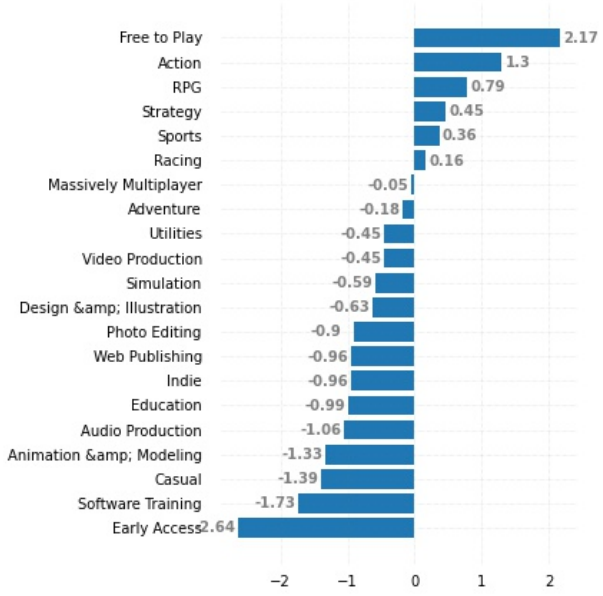

Fig. 5: Values of learned β_g in our model

We also show analysis on the learned parameters in our model. In Fig. 5 we show the values of learned β_g. This parameter covers the bias of each genre. It stands for how likely that a user would buy a game if the game has this genre. The results are in line with our experience. A user would obviously like to "buy" a Free-to-play game. Other popular genres like Action, RPG, Strategy all have positive effects. Whereas some genres like Education, Audio Production, Software Training are not popular. And users are least likely to buy Early Access games.

We now discuss some alternations we haven't tried. Although we are satisfied with its current performance, we presume that several improvements and alternation could be added to our model. Similar to genres, information of game developers, prices, discounts can also be utilized. What's more, the way of representing a genre set can also be discussed. A Multi-layer Perceptron could have the potential to surpass the method of linearly combining the genres. However, it would be easier to overfit and needs careful tuning in hyperparameters to balance between representational power and robustness.

VII. Conclusion

In this paper we have built a recommender system to address the item cold start problems on *Steam* dataset. We first did analysis on this dataset. Then we show why a Collaborative Filtering Model would not work on item cold start scenarios. To address this issue, we built a model that utilize genre information so that it may predict unseen items with no difficulty. We further build a dataset specifically for item cold start problems. By evaluating our model on this dataset, we demonstrate its advantages over the alternative methods.

References

[1] M. Wan and J. McAuley, "Item recommendation on monotonic behavior chains," in *Proceedings of the 12th ACM Conference on Recommender Systems*, ser. RecSys '18, Vancouver, British Columbia, Canada: Association for Computing Machinery, 2018, pp. 86–94, ISBN: 9781450359016. DOI: 10 . 1145 / 3240323 . 3240369. [Online]. Available: https://doi.org/10.1145/3240323.3240369.

[2] A. Pathak, K. Gupta, and J. McAuley, "Generating and personalizing bundle recommendations on steam," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, pp. 1073–1076.

[3] W. Kang and J. McAuley, "Self-attentive sequential recommendation," in *2018 IEEE International Conference on Data Mining (ICDM)*, 2018, pp. 197–206. DOI: 10.1109/ICDM.2018.00035.

[4] X. Li and B. Zhang, "A preliminary network analysis on steam game tags: Another way of understanding game genres,"

[5] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," *arXiv preprint arXiv:1205.2618*, 2012.

[6] R. M. Bell, Y. Koren, and C. Volinsky, "The bellkor 2008 solution to the netflix prize," *Statistics Research Department at AT&T Research*, vol. 1, 2008.

[7] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[8] T.-W. Sung, "How to rank #1 in task 1,piazza cse258 note348,"

[9] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 426–434.

[10] S.-T. Park and W. Chu, "Pairwise preference regression for cold-start recommendation," in *Proceedings of the third ACM conference on Recommender systems*, 2009, pp. 21–28.

[11] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl, "Getting to know you: Learning new user preferences in recommender systems," in *Proceedings of the 7th international conference on Intelligent user interfaces*, 2002, pp. 127–134.

[12] K. Zhou, S.-H. Yang, and H. Zha, "Functional matrix factorizations for cold-start recommendation," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, 2011, pp. 315–324.