Can Item Metadata help combat the Cold Start problem?

Abstract

The cold start problem in recommender systems arises when dealing with new items with no prior interaction history, making it challenging to generate accurate recommendations. This work explores whether incorporating item metadata can effectively address this problem in game recommendations using user-game interactions on the Steam platform. We leverage game metadata (genres and price) alongside implicit feedback from user playtime data, comparing five different recommendation approaches: traditional Matrix Factorization (SVD), Factorization Machines (FM), hybrid methods combining SVD/FM with K-Nearest Neighbors (KNN), and a metadata-only approach using FM. An analysis of results show that the SVD baseline shows strong performance on existing games but degrades for new games, demonstrating the cold start problem. Surprisingly, incorporating metadata through FMs doesn't improve performance, suggesting that with the Steam dataset's scale (>5 million interactions), the collaborative filtering signal alone may be sufficiently strong. The hybrid method which incorporate game metadata (Method III) show significant improvements on items prone to cold start problems through extending their base counterparts, while the metadata-only approach (Method V) performs similarly to Method II and IV, indicating that in this context, item metadata provides limited additional signal beyond user-game interactions.

Keywords

Recommender Systems, Factorization Machines, Implicit Feedback

1 Introduction

The "cold start" problem is an issue in recommender systems which deals with how to work with recommendations on items or users that have minimal or no interactions. As this problem is often not handled by standard recommender algorithms such as Collaborative Filtering models or Factorization Machines due to the fixed user-item matrix constructed during training, there needs to be alternatives to ensure new users or items still get robust recommendations using whatever information is available. This work aims to explore the use of standardized recommender system algorithms in conjunction with item metadata to combat the cold start problem.

2 Dataset

This work uses the Steam User-Game dataset used in [7, 11, 13] consisting of user-game interactions and game metadata obtained from the Steam Web API.

2.1 User-Game Dataset

This dataset consists of user-game interactions in terms of the overall hours played. The format of the data used is illustrated in Table 1 and some of its basic statistics of the data is presented in Table 2. Due to the absence of a explicit rating field in the dataset and also on the Steam platform (besides a Thumbs Up/Down option), this work decides to explore the use of a users' playtime as an implicit measure of how much they might like/dislike a game.

Table 1: Dataset Format

steam_id (User)	item_id (Game)	playtime_forever
76561197970xxxxxx	20	6
76561197970xxxxxx	300	4733
76561198329xxxxxx	304930	677

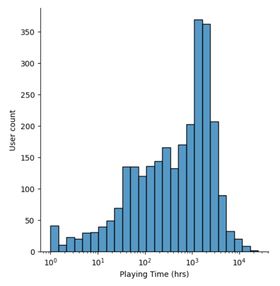
Observing Table 2, it is evident from the difference in the min and max and the mean being much greater than median of purchases per item, that data is right-skewed, implying some items are very popular in comparison to others. This skew is also evident in purchases per user, implying some users purchase way more than in comparison. However, since these behaviours cannot be considered abnormal, corrections are not made to these skews.

Another interesting statistic that is present in the dataset is 1,867,963 of the purchases (approximately 36% of all purchases) result in no further interaction, i.e, user purchases a game and does not play it. Since this behaviour could be due to various reasons such as the game being free, bought during a sale, gifting, etc..., and since these behaviours do not necessarily imply that the user does and will not like this game, these rows are dropped from further analyses.

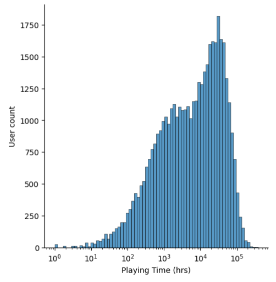
Table 2: Dataset Statistics

Statistic	Value
Total Users	88,310
Users who purchased at least one game	71,504 (80.9%)
Total Games	10,978
Total Purchases	5,153,209
Min (non-zero), Max purchases per user	1; 7762
Mean, Median purchases per user	58.35; 26
Min, Max purchases per item	1; 49,571
Mean, Median purchases per item	469.41; 43

Further analysis is carried out on the playtime data due to it being the "field of interest". It comes as no surprise that the range of playtime ranges from 1 to approx 6×10^6 hours. The extremely large playtimes hours can be explained by online games or non-story based games such as CS:GO, League of Legends which tend to have much more repeat players in comparsion to story based games. This is illustrated with two specific examples in Figure 1 where the maximum playtime for the story-based game is around 10^4 hours compared to a large percentage of playtimes for the online game lying between 10^4 and 10^5 hours. This difference in playtimes ranges for varying game types and the extreme distribution of values irrespective of the type of game (illustrated in Figure 2), necessitates pre-processing before modelling.



(a) Assassin's Creed II (Story-based)



(b) CS:GO (Online)

Figure 1: Log-scale playtime distribution

Pre-processing of the data is carried out in multiple steps:

(1) Initially, extremely large playtime hours which can be considered outliers are removed using the Inter-Quartile Range (IQR) and Quartile 3 ($Q3$) of each game's playtime if,

$$playtime\_forever > Q3 + 1.5 * IQR$$

(2) Due to the extreme range of playtime hours and the a normal-like distribution on the log-scale plots, log-based normalization is applied to the data next.

(3) Due to the uniform range preferred for rating prediction in recommender system algorithms, we pre-process the data for each game further using min-max scaling to $[1, 5]$ using the min and max playtimes of that game. A global min-max scaling for all games is avoided due to the extremely varying playtime ranges present for different games.

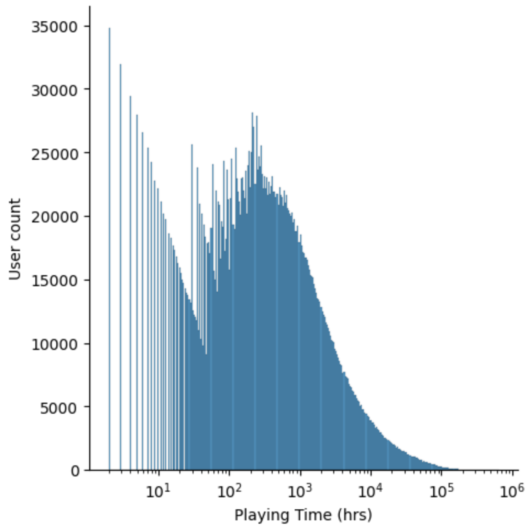


Figure 2: Log-scale playtime distribution for all-games

The distribution of the pre-processed data is visualized in Figure 3. This data which appears almost normally-distributed will be used in further modelling for recommendation of games to users.

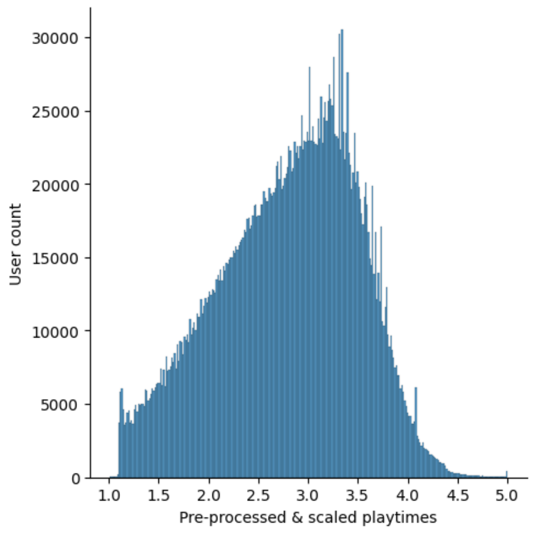


Figure 3: Pre-processed distribution for all-games

For the data split into training and testing sets, we use the fact that Steam game IDs are **numbered sequentially in order of release**. Using this, we split the data into three sets as follows:

- $item\_id <= 300,000$: Consists of approximately 80% of user-game interactions. This data is further split randomly using a 80-20 split into **Train Set** and **Test Set I**. These sets are used as a baseline which is used for training standard recommender models. In these sets, since the testing is done on a subset of the same overall set, it does not have to deal with unseen games or users (except those maybe split during random shuffle).
- $item\_id > 300,000$: **Test Set II** which consists of the remaining 20% of user-game interactions. The difference of this set from Test Set I is the fact that this set consists of newer items due to the split by sequential $item\_id$, i.e, these games are unseen during training and can be considered "new games" (referred to as so hereon) as far the model trained on the previous set is concerned. Thus, data in this set is prone to the cold start problem and is used for evaluating our methods which explore solving this problem.

2.2 Game Metadata Dataset

This dataset is used to explore addressing the cold start problem. Due to the user-game interactions only containing game and user IDs, this additional information is obtained through the official Steam Web API. While a lot of metadata is made available through this API, the data we collect which seemed to be of interest are described in Table 3.

Of the 10,978 games in the interactions dataset, metadata was not available on the official Web API for approximately 700 of the

Table 3: Dataset Format

item_id (Game)	Genres	Price ($)
20	[Action, Indie]	12.99
300	[Survival, Shooting]	5.49
304930	[Adventure, Free to Play]	0.00

games. Interactions on these games are dropped in calculations for the models involving requiring it.

3 Predictive Task

Given the user-game interaction and the metadata of each game, our task is to predict the "scaled playtime" of a user for a game. We consider the pre-processed user-game interaction data scaled from 1 to 5 as a scale of how much a user plays a game compared to the average playtime for the game, which can in turn be thought of as an implicit way to think about how much a user might like a game which makes it more likely to be a better recommendation. This claim makes it a good fit as a "rating prediction" task using recommender system algorithms. Hereon, the scaled playtime variable is represented using $r_{u,i}$ and referred to as "rating" for ease of understanding.

The tasks are evaluated using Mean-Squared Error (MSE) as follows:

$$MSE = (\hat{r_{u,i}} - r_{u,i})^2 \tag{1}$$

4 Related Work

Although we discuss related work throughout our study, this section provides a more concise and focused view of the ongoing works that have motivated our methodologies, informed our research, and shaped the models we employed in this project.

Hyperparameter Optimization for Recommender Systems

Bergstra and Bengio (2012): *Random Search for Hyper-Parameter Optimization* [2] This paper's focus is on demonstrating how to effectively get the best hyper-parameter values for tuning models. The practice they describe, 'grid search' is used throughout our models in tuning hyperparameters such as the number of neighbors considered for KNNs or the number of latent factors in our SVD models. The search strategy helps us efficiently explore the parameters space, getting the best yield from our models.

Collaborative Filtering with Neighborhood Models

Koren (2010): *Factor in the Neighbors: Scalable and Accurate Collaborative Filtering* [8] This study describes how to make the baseline KNN estimator model that we used in our studies. Although prior works have been done on KNN models, this paper specifically focuses on enhancing baseline estimates for the cold start problem. This approach served as the cornerstone for our integration of KNN with baseline estimators.

Metadata-Driven Enhancements to Recommender Systems

Frémal and Lecron (2017): The research *Weighting Strategies for a Recommender System Using Item Clustering Based on Genres* [3] explored the use of item metadata, such as genres, to improve recommendation accuracy. Inspired by this approach, we integrated metadata-driven enhancements with latent factor models to examine whether additional contextual information could refine recommendations and bolster performance.

Singular Value Decomposition in Collaborative Filtering

Funk (2006): The blog *Netflix Update: Try This at Home* [4] by Simon Funk popularized the use of SVD for collaborative filtering, laying the groundwork for scalable and accurate recommender systems. The Surprise library's SVD implementation, which is based on Funk's algorithm, played a central role in our study. Multiple models in our experiments relied on SVD, underscoring its importance in achieving strong performance across both seen and unseen data.

These works collectively informed our methodological framework and experimental design, providing both theoretical and practical insights that enabled the development of robust hybrid models. Our study builds upon and extends these contributions, particularly in addressing challenges related to cold start scenarios and optimizing hybrid approaches for diverse datasets.

5 Models

To compare performance across different methods, five different recommender system models are used in the rating prediction. Methods I & II which use factorization techniques are used as baselines as they do not have the capability to combat the cold start problem. Methods III & IV extend Methods I & II to incorporate some mechanisms to combat the cold start problem using item metadata. Method V uses a factorization technique solely on item metadata technique taking *item_id* out of the picture completely. The methods are detailed as follows:

5.1 Method I: Collaborative Filtering - SVD

A standard matrix factorization algorithm [9] used for rating prediction in recommender systems is used as a baseline. The gist of this method assumes that if there are many commonalities or patterns in user-item interactions, these interactions can be approximated as the product of two lower-rank factorization terms, one for user and one for item. The low-rank value k which may represent these patterns are known as latent factors.

This generalization allows us to predict values of unseen user-item interactions by approximating the factorization terms using seen user-item interactions. The model prediction equation is as follows:

$$\hat{r_{u,i}} = \mu + \beta_u + \beta_i + \gamma_u . \gamma_i \tag{2}$$

where $\hat{r_{u,i}}$ is the predicted rating, μ is the common bias term, β_u and β_i are the bias terms for users and games respectively and γ_u

γ_i are the latent factor matrix factorization term. During train, we minimize the MSE loss with a regularization term as follows:

$$\mathcal{L} = (\hat{r_{u,i}} - r_{u,i})^2 + \lambda({\beta_u}^2 + {\beta_i}^2 + ||\gamma_u||^2 + ||\gamma_i||^2) \quad (3)$$

where λ is the regularization factor.

5.2 Method II: Factorization Machine (FM)

Factorization Machines (FM) [12] extend traditional matrix factorization by modeling all possible pairwise interactions between features using factorized parameters. Unlike SVD which can only handle user-item interactions, FMs can incorporate additional features like item metadata. The model prediction equation for a FM of degree d=2 is:

$$\hat{y}(\mathbf{x}) = w0 + \sum i = 1^n w i x_i + \sum i = 1^n \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j \quad (4)$$

where w_0 is the global bias, w_i are linear weights for each feature, and $\mathbf{v}_i, \mathbf{v}j \in \mathbb{R}^k$ are k-dimensional factorization vectors that model the pairwise interactions between features x_i and x_j. The feature vector used in this method for each user-game interaction consists of:

- One-hot encoded user IDs
- One-hot encoded game IDs
- Game genres (multi-hot encoded)
- Normalized price
- Scaled playtime

During training, we minimize the MSE loss with a regularization term similar to Method I:

$$\mathcal{L} = (\hat{y}(\mathbf{x}) - y)^2 + \lambda_w \sum i = 0^n w i^2 + \lambda_v \sum i = 1^n |\mathbf{v}_i|^2 \quad (5)$$

where λ_w and λ_v are regularization parameters.

5.3 Method III (A): KNN Method with Baseline Estimator

This model is primarily a build-up to Method III, which combines the strongest aspects of the KNN baseline and SVD models. The KNN model aggregates information from a pre-defined number of neighbors, k. Since there is no one-size-fits-all solution for KNN algorithms, we specified a k value using random and grid search techniques (Bergstra and Bengio, 2012) [2]. To increase accuracy, we incorporated baseline estimates as defined in Formula 3, Section 2.2 of (Koren, 2010) [8].

$$r_{ui} = b_{ui} + \frac{\sum_{j \in N_u^k(i)} \text{sim}(i,j) \cdot (r_{uj} - b_{uj})}{\sum_{j \in N_u^k(i)} \text{sim}(i,j)}$$

where $\text{sim}(i,j)$ is the similarity between items i and j, defined using the Mean Squared Difference (MSD) formula:

$$\text{sim}(i,j) = \frac{\sum_{u \in U}(r_{ui} - \hat{r}_{ui})(r_{uj} - \hat{r}_{uj})}{\sqrt{\sum_{u \in U}(r_{ui} - \hat{r}_{ui})^2 \cdot \sum_{u \in U}(r_{uj} - \hat{r}_{uj})^2}}$$

Limitations. Due to a large dataset, memory limitations arose on local machines and Google Colab. For $k > 5$, the training size had to be reduced, affecting model robustness. This also prevented experimenting with alternative approaches, such as embedding metadata in vector representations.

5.4 Method III: SVD + KNN

After noticing that the SVD model outperforms KNN on test set I but deteriorates on test set II, whereas the KNN model outperforms the SVD model on test set II, we leverage the strengths of both models to form this hybrid model.

This custom model integrates their complementary capabilities to address key challenges in recommendation systems. Specifically, it employs the SVD prediction algorithm for interactions with previously seen items, capitalizing on its superior performance in modeling latent factors. Simultaneously, the model utilizes KNN's baseline estimator to predict ratings for unseen items, combating the cold start problem by extrapolating preferences based on similar users or items. This approach allows the model to balance precision in known contexts with the handling of new and sparse data scenarios.

5.5 Method IV: Factorization Machine + KNN

Given the good performance of FastFM ALS and its limitation in solving the cold start problem, it is worthwhile to test the power of KNN combined with the FastFM ALS model. The plan was to run the KNN used in Method III(A) on all the training data and test it on the newer games to see its improvement. Due to the intrinsic large size of the sparse matrix used in FastFM ALS, the memory left for KNN is small, so this model is hard to get scaled. The final training set reached 30000 samples with 3 nearest neighbors on Google Colab.

5.6 Method V: Factorization Machine - Game Metadata Only

This method is an extension of Method II and it explores the feasibility of a factorization machine using only the game metadata in the rating prediction task. This method is different from all previous methods in the fact that the factorization terms in this method used do not contain the *game_id* information but rather tries to build a predictor based on user interactions with the metadata of the game. The *genres* and *price* of the game are the metadata used in this model. This work is similar to the method explored by [3] which builds latent-factor models for each genre on the MovieLens dataset [5].

However, the fact that *game_id* is ignored means this model should be more robust to cold start problems due to the sole dependence of game metadata which is available for newer games as well.

6 Results

The results across models described in Section 5 are presented in Table 4. Further elaboration and discussion of results for each method is presented below.

Table 4: Results

Method	MSE (Test Set I)	MSE (Test Set II)
I (SVD)	0.3275	0.5684
II (FM)	0.4471	0.5821
III(A) (KNN)	0.4063	0.4397
III (SVD+KNN)	**0.2855**	**0.2916**
IV (FM+KNN)	-	0.5884
V (FM-Metadata)	0.4378	0.5656

6.1 Method I

Method I implements the SVD factorization method of the Scikit-Surprise library [6]. The optimal parameters are found using Grid Search with the number of latent factors being set 5 and number of epochs as 25 while all other parameters following default values from the library. The low number of latent factors implies the patterns among user-games are not as varied. The MSE of this method on the test sets is used as a baseline to compare the proposed methods. It can be noted that the MSE on Test Set II is significantly higher than that on Test Set I. This behaviour is expected since Test Set II consists of new games prone to the cold start problem which this method handles solely by giving the user's average rating which is not necessarily a good heuristic.

6.2 Method II

Method II implements the ALS Regression method of the fastFM library [1]. The model was trained with $rank = 10$ and $n\_iter = 50$ factors. From Table 4, we can observe that the Factorization Machine model achieves an MSE of 0.4471 on Test Set I and 0.5821 on Test Set II. The higher MSE compared to the SVD baseline (Method I) suggests that the additional complexity of modeling feature interactions through factorization machines did not lead to improved performance in this case. This could be attributed to the large scale of the Steam dataset - with over 5 million user-item interactions, the collaborative filtering signal (user-item interactions) appears to be strong enough to create meaningful latent representations on its own. In this scenario, the additional item metadata (genres, price, and playtime) might actually introduce noise rather than valuable signal, suggesting that when the interaction data is sufficiently rich, simple collaborative filtering approaches may be more effective than hybrid methods that incorporate content features.

6.3 Method III(A)

The KNN model performed worse than SVD (Method I), as expected. SVD scales better with increased data sizes (Kabić et al., 2020) [10] because the number of computations does not grow as rapidly as in KNN. For KNN, each new training item requires at least k new comparisons, increasing computational cost significantly. The optimal value is chosen as $k = 3$.

6.4 Method III

Unsurprisingly, this hybrid model outperforms both the standalone SVD and KNN baselines. By integrating KNN's baseline estimator for new user/item predictions with SVD's scalable and accurate handling of seen interactions, the model demonstrates superior overall performance and shows resilience in addressing cold start challenges.

This is evident when comparing the MSE across Test Set I (containing mostly seen interactions) and Test Set II (with previously unseen interactions). While the hybrid model performs better than standalone SVD on Test Set II, the differences in MSE are not substantial, suggesting that the contribution of KNN to mitigating the cold start issue is relatively modest. This indicates that the hybridization, while helpful, does not dramatically enhance performance over SVD alone.

Nevertheless, the hybrid approach still outperforms the pure KNN baseline both in accuracy and computational efficiency, particularly on unseen games in Test Set II. These results validate the model's ability to balance accuracy and adaptability, though the extent of improvement over SVD in handling cold start issues is less pronounced than anticipated. This finding highlights that while hybrid models offer potential, the incremental gains may vary depending on the data and context.

6.5 Method IV

The final training set reached 30000 samples with K = 3 on Google Colab and the improvement in MSE was limited, as shown in Table 4. Larger training set enhances the performance of the model and we expect further improvement of the model if run on larger memory.

To demonstrate the effectiveness of this model in ameliorating the cold start problem, the ALS model and the ALS + KNN model were run on small subsets of the training set. Table 5 shows the improvement of the model run on these subsets.

Table 5: MSEs of model II and model IV on a subset of the training set

Set	MSE (ALS)	MSE (ALS + KNN)
1	1.4004	0.5892
2	1.9474	0.5924

6.6 Method V

Method V implements the ALS Regression method of the fastFM library [1]. The optimal parameters are found to be number of latent factors as 2 and epochs as 100. The extremely low latent factor count is unexpected due to the numerous count of genre combinations, but as seen in previous methods, the patterns might not be complex, i.e, it can be something as simple as a high-price low-price differentiation or story-based or online differentation which can still be represented using very few factors. This behavior is also seen to be evident on rating prediction in the GoodReads dataset [13]. From Table 4, it is noticed that the MSE for this method is worse than other methods for Test Set I. This can be explained due to the replacement of the user-game interactions in favour of user-metadata interactions. However, this significant difference in MSE on Test Set I suggests that this method is not worth the slight improvement noted in MSE on Test Set II in comparison to the baseline.

7 Conclusion & Future Work

Our study provides several key insights for recommender system design in large-scale commercial settings:

- When interaction data is abundant (>5 million interactions in our case), simple collaborative filtering approaches can outperform more complex hybrid methods as seen in Method I vs Method II and V. This suggests that the complexity of incorporating metadata features may not always justify the implementation overhead.
- Item metadata shows more promise in true cold start scenarios as seen in Method III which performs significantly better on new games on Test Set II as compared to other models which have no mechanisms to handle the cold start problems.
- The trade-off between model complexity and performance should be carefully considered based on the specific application context. Our results show that simpler models (SVD) achieved lower MSE compared to more complex approaches (Factorization Machines with metadata), which challenges the idea that more features necessarily lead to better recommendations.

These findings extend beyond game recommendations and provide valuable insights for the broader recommender systems community. The counterintuitive result that metadata features do not always introduce more signal than noise as evidenced in Methods II, IV and V in large-scale systems challenges common assumptions about hybrid recommender systems. However, as seen in Method III, metadata can be informative in certain scenarios as well. Thus, the exploration of this work on the use of item metadata for recommendations concludes that its impact remains highly dependent on the type of interactions, the dataset, the category of metadata and the models used.

7.1 Future Directions

Several promising directions emerge from our findings:

- Investigation of dynamic feature importance based on data scale - understanding when metadata becomes less valuable as interaction data grows
- Development of adaptive hybrid models that can automatically adjust the weight of metadata features based on interaction data availability
- Exploration of more sophisticated metadata incorporation techniques, such as attention mechanisms or graph-based approaches

References

[1] Immanuel Bayer. 2016. fastFM: A Library for Factorization Machines. *Journal of Machine Learning Research* 17, 184 (2016), 1–5. http://jmlr.org/papers/v17/15-355.html

[2] James Bergstra and Yoshua Bengio. 2012. Random Search for Hyper-Parameter Optimization. In *The Journal of Machine Learning Research, 13(1), 281-30.*

[3] Sébastien Frémal and Fabian Lecron. 2017. Weighting strategies for a recommender system using item clustering based on genres. *Expert Systems with Applications* 77 (2017), 105–113. https://doi.org/10.1016/j.eswa.2017.01.031

[4] Simon Funk. 2006. Netflix update: Try this at home. https://sifter.org/~simon/journal/20061211.html

[5] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (Dec. 2015), 19 pages. https://doi.org/10.1145/2827872

[6] Nicolas Hug. 2020. Surprise: A Python library for recommender systems. *Journal of Open Source Software* 5, 52 (2020), 2174. https://doi.org/10.21105/joss.02174

[7] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM).* IEEE, 197–206.

[8] Yehuda Koren. 2010. Factor in the neighbors: Scalable and accurate collaborative filtering. In *ACM Transactions on Knowledge Discovery from Data (TKDD), 4(1), 1-24.*

[9] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37. https://doi.org/10.1109/MC.2009.263

[10] Gabriel Duque López Marko Kabić and Daniel Keller. 2020. A Refined SVD Algorithm for Collaborative Filtering. In *arXiv preprint arXiv:2012.06923.*

[11] Apurva Pathak, Kshitiz Gupta, and Julian McAuley. 2017. Generating and personalizing bundle recommendations on steam. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 1073–1076.

[12] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000. https://doi.org/10.1109/ICDM.2010.127

[13] Mengting Wan and Julian McAuley. 2018. Item recommendation on monotonic behavior chains. In *Proceedings of the 12th ACM conference on recommender systems.* 86–94.