From Counter-intuitive Observations to a Fresh Look at Recommender System

AIXIN SUN, Nanyang Technological University, Singapore

Recently, a few papers report counter-intuitive observations made from experiments on recommender system (RecSys), both in offline and online settings. One observation is that users who spend more time and users who have many interactions with a recommendation system receive poorer recommendations, compared to users who spend less time or who have relatively fewer interactions. Another observation is that models trained by using only the more recent parts of a dataset show significant performance improvement. In this opinion paper, we interpret these counter-intuitive observations from two perspectives. First, the observations are made with respect to the global timeline of user-item interactions. Second, the observations are considered counter-intuitive because they contradict our expectation on a recommender, i.e., the more interactions a user has, the higher chance that the recommender better learns the user’s preference.

For the first perspective, we discuss the importance of the global timeline by using the simplest baseline Popularity as a starting point. We try to answer two questions: (i) why the simplest model popularity is often ill-defined in academic research? and (ii) why the popularity baseline is evaluated in this way? The answers lead to the next question: If the simplest baseline is not evaluated in a proper manner, what about the other more complex models? This question leads to a detailed discussion on the data leakage issue that exists in many offline evaluations. As the result, model accuracies reported in many academic papers are less meaningful and incomparable. For the second perspective, we try to answer two more questions: (i) why models trained by using only the more recent parts of data demonstrate better performance? and (ii) why more interactions from users lead to poorer recommendations? The key to both questions is user preference modeling. Based on the discussion, we propose to have a fresh look at RecSys. The problem definition remains the same i.e., to learn from users’ past interactions and then recommend users unseen items of interest. However, the evaluation in the offline setting must observe the global timeline i.e., considering the absolute time points of user-item interactions in chronological order. We then discuss how to conduct more practical offline evaluations and possible ways to effectively model user preferences. The discussion and opinions in this paper are on top-N recommendation only, not on rating prediction.

CCS Concepts: • Information systems → Recommender systems; Collaborative filtering.

Additional Key Words and Phrases: Recommender System, Global Timeline, Practical Evaluation, User Preference Modeling

ACM Reference Format:
Aixin Sun. 2022. From Counter-intuitive Observations to a Fresh Look at Recommender System. J. ACM xx, x, Article 111 (October 2022), 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Recently, a few papers report counter-intuitive observations made from experiments on recommender system (RecSys), both in offline and online settings [13, 18, 27, 28, 30]. For example, Ji et al. [13] report that both users who spend more time and users who have many interactions with a...
recommendation system receive poorer recommendations, compared to users who spend less time or who have relatively fewer interactions with the system. This observation holds on recommendation results by multiple models (i.e., BPR [24], Neural MF [10], LightGCN [9], SASRec [14] and TiSASRec [17]) on multiple datasets including MovieLens-25M, Yelp, Amazon-music, and Amazon-electronic. On a large Internet footwear vendor, through online experiments, Sysko-Romaniczuk et al. [27] observe that “experience with the vendor showed a negative correlation with recommendation performance”. The factors considered under “experience” include the number of days since account creation and since the first shopping transaction respectively, and the number and the value of purchase transactions made in the past year. A recent study also reports that “using only the more recent parts of a dataset can drastically improve the performance of a recommendation system” [30]. In this opinion paper, we interpret the reported counter-intuitive observations from two perspectives.

First, these observations are made with respect to the time dimension, more specifically, the global timeline of user-item interactions. Here, we are not considering time as an additional feature or context in the modeling. Rather, we consider the arrangement of the user-item interactions in chronological order along timeline. Hence, we have “number of days since the first transaction” and “recent parts of a dataset”. These observations cannot be made if we do not arrange user-item interactions along the global timeline chronologically. Although there are models which consider time as a contextual feature in their modeling, not many studies arrange user-item interactions along the global timeline chronologically, and consider the absolute time points of the interactions. The reported counter-intuitive observations call for a revisit of the importance of observing the global timeline. Findings from the revisit may impact our model design and evaluation, and more importantly, our understanding of recommender system. To be discussed shortly, because of the missing of timeline, even the simplest recommendation model, Popularity, is ill-defined in many RecSys papers. This leads to the first and relatively simple question: why the simplest model Popularity is often ill-defined in academic research? If an ill-defined model is used as a baseline in comparison, are the other models properly evaluated? The answer is no. As the result, model accuracies reported in many papers are less meaningful and not comparable, due to the ignorance of the global timeline.

Second, these observations are considered counter-intuitive because they contradict our expectation (or an implicit assumption) on a recommender. That is, the more interactions a user has with a system, the higher chance that the recommender better learns the user’s preference. However, these observations show that more interactions do not lead to better user preference learning. Does the user preference learned through user-item interactions indeed reflect the kind of user preference in our expectation? To answer this question, we may need to revisit what user preference is.

In the following, we start with the simplest baseline, i.e., Popularity, and discuss why this commonly used baseline is ill-defined.

2 ILL-DEFINED POPULARITY AND DATA LEAKAGE

As a non-personalized method, Popularity is often considered as the simplest recommendation baseline and is widely used for comparison purpose in evaluation. In many academic papers, the popularity of an item is defined as the number of interactions it receives in the training data. An interesting question here: Is the popularity defined by item frequency in training set even meaningful?

To answer this question, we first show how a recommender works in general. With the help of a global timeline, Figure 1 gives absolute time points of historical user-item interactions for three example users \(u_1\), \(u_2\), and \(u_3\). In the illustration, \(t_c\) indicates the current time point. If the users visit the website at the current time \(t_c\), then the system will make recommendations to these users. Users then may choose to interact or not to interact with the recommended items. In the illustration, let
we assume that all users interact with the recommended items, and these interacted items become the latest interactions for all three users. In practice, a recommender can learn from all or a subset of historical interactions that occurred before time $t_c$, and makes recommendations if users visit the site at time $t_c$.

Now, let us look at some real-world examples of popularity ranking. The New York Times best sellers\footnote{https://www.nytimes.com/books/best-sellers/} is a well-known and influential list of best-selling books in the United States. The list is updated on a \textit{weekly basis} since 1931.\footnote{https://en.wikipedia.org/wiki/The_New_York_Times_Best_Seller_list} There is also an option to display \textit{monthly} lists of different book genres. The best sellers on Amazon is an hourly updated list.\footnote{https://www.amazon.com/bestsellers} It is common for websites to have a popularity ranking of items on an hourly, daily, weekly, or monthly basis.

The real-world popularity rankings have two important properties. First, the rankings are dynamically updated along timeline. For instance, the best sellers on the New York Times in Week 1 of Year 2020 shall be quite different from that in Week 3 of Year 2022. Second, the popularity ranking only considers item frequency in a predefined time range (e.g., an hour, a week, or a month), and does not necessarily use all historical data. The current weekly ranking only needs to use the interactions occurred in the past week, and the current hourly ranking only use interactions in the past hour.

The popularity baseline used in many RecSys offline evaluations is very different. We do not define a time window (e.g., a week or a month), and we force the popularity to use all interactions in training set. In other words, the frequency-based ranking is a \textit{static} ranking, and there is only one big time window, \textit{i.e.}, the \textit{entire duration of the training data} which is determined by the adopted data partition scheme. \textit{Is this popularity baseline even meaningful?} No. In the following, let us map the popularity baseline to a timeline.

Most academic researchers do not have access to an online platform to directly evaluate their models by real user-item interactions. Evaluation on an offline dataset is the only choice in most cases. For offline evaluation, we typically partition a dataset into two sets, training set and test set. Without the actual test instances (\textit{i.e.}, the events that users visit the system), we sample some user-item interactions as test instances. There are many ways of sampling test instances, also known as dataset partition schemes \cite{20, 26, 34}. For example, test instances can be randomly sampled from the offline dataset, and the remaining interactions form the training set. A widely used scheme is \textit{leave-last-one-out}, or simply \textit{leave-one-out}. That is, the last interaction of each user is sampled as a test instance, and the user’s remaining interactions are in the training set.

Figure 2 provides an illustration of the leave-one-out data partition for three example users, with respect to the global timeline. Observe that the test instance for \textit{u}_1 occurs at time $t_{x1}$. If we
consider time $t_{x1}$ to be the current time $t_c$ as if the offline evaluation were online, then all the historical interactions a recommender can learn from at $t_{x1}$ should be the three interactions by $u_1$ and the one interaction by $u_2$. A recommender would never have access to the future interactions that will happen in the future with respect to time point $t_{x1}$. At $t_{x1}$, the future items include two interactions by $u_3$ and all interactions by user $u_3$, which occur after $t_{x1}$. By forcing the popularity baseline to use all training data, the popularity method may recommend some items to $u_1$ that are very popular in future, with respect to time $t_{x1}$. Clearly, the popularity baseline in many offline evaluations are ill-defined, and not meaningful. If we follow the popularity definition in real world, the performance of popularity increases by at least 70% compared to the ill-defined popularity on MovieLens dataset [11].

This leads to the next question: Why the popularity baseline in academic research is evaluated in this way? The reason is simple. We want to ensure a “fair comparison”, where all models are expected to use the same training set. Basically, our machine learning- or deep learning-based models are trained on the training set and evaluated on the test set. The popularity baseline is treated as a trivial machine learning model. It takes in all instances in the training set and produces a ranking by the ill-defined popularity for the purpose of “fair comparion” with others. Then, if the simplest baseline is not evaluated in a proper manner, what about the other more complex models? The answer is no as well.

Unfortunately, due to the leave-one-out data partition scheme, all offline models suffer from the same issue as the ill-defined popularity: accessing future data that is impossible to access in reality. Formally, this is known as data leakage in machine learning, and Ji et al. [12] offers a detailed study on this topic in the context of RecSys. From this perspective, the more complex models may not be evaluated in a practical manner as they access future data, unless the data partition scheme leads to no (or at least minimum) data leakage. For example, if the dataset is partitioned by an absolute time point (i.e., all interactions occurred before $t_j$ are training data, after are test data), then there is no data leakage. Reported in [12], a major impact of data leakage is that the model accuracies reported in the literature is incomparable.

In summary, the evaluations conducted without observing the global timeline in an offline setting may suffer from data leakage, rendering to incomparable results. Experiments conducted in this way also contribute to the difficulty of reproducibility. In reality, users may interact with a system at any time; new items may become available in the system at any time; outdated item are removed from the system at any time. For example, an iPhone model is usually discontinued after two years of its release, and the phone changes from its first generation in 2007 to iPhone 14 in 2022. Many widely used datasets in RecSys research cover user-item interactions collected in a long time period, e.g., more than 10 years for MovieLens, Yelp, and Amazon [12]. A simple way of partitioning data into train/test without considering timeline is a much simplification to the research on RecSys.
Fig. 3. Context for decision making of an interaction e.g., phone purchase

With time taken into consideration in recent evaluations [13, 15, 16, 30], we start to obtain counter-intuitive observations listed in the Introduction section. The discussion so far on the ignorance of global timeline does not explain (i) why models trained by using only the more recent parts of data demonstrate better performance? and (ii) why more interactions from users lead to poorer recommendations? A hypothetical answer to both questions is the simplification in learning user preference in current models.

3 SIMPLIFICATION IN USER PREFERENCE MODELING

The missing of the global timeline could be a contributing factor to our misunderstanding in user preference modeling. To better understand user preference modeling, let us go back 30 years to learn how collaborative filtering was firstly defined in the Tapestry system [8].

The way that Tapestry system supports collaborative filtering is to let users to read documents recommended by other users. The authors give an example in their original paper [8]. A user $u$ wants to read interesting but not all documents from a newsgroup. She knows that some users read all of these documents and mark the interesting ones. She then can simply choose to read only the documents that are marked interesting by these users. Such kind of filtering is conceptually similar to reading only the tweets written or retweeted by the users one follows on Twitter.

In Tapestry (and similarly Twitter), user preference is directly reflected by the other users he/she follows. A hypothetical extension of the understanding is that if user $u_1$ follows $u_2$, then $u_1$ prefers $u_2$’s decision making in judging interesting documents (or retweeting) given the context at that time, e.g., when a document is received in the newsgroup.

Different from Tapestry, where users are empowered to choose who to follow, user preference in mainstream RecSys research is inferred from user-item interactions. The main underlying assumption is that a user $u$ would prefer the items that are chosen by other users who share similar preferences with $u$. Preference similarity between users is reflected by similar user-item interactions in the past. If users $u_1$ and $u_2$ both purchased the same mobile phone, then we would consider that $u_1$ and $u_2$ share similar preference, at least on this particular item. However, purchasing the same phone may not necessarily reflect that the two users share a similar decision making process, if we consider the context changes in a system from time to time.

Figure 3 shows an example scenario where the three users $u_1$, $u_2$, and $u_3$ purchased the same phone at different time points, $t_1$, $t_2$, and $t_3$ respectively. We may further consider that $t_1$ is the first day when this phone was released, and $t_3$ is among the last few days when this phone was to be discontinued, and $t_2$ is the middle time in between. We may also consider that an upgraded version of this phone has been released in between $t_2$ and $t_3$. In this scenario, the three decision makings could be very different, because the alternative phone models to choose from at the three time points $t_1$, $t_2$, and $t_3$ will be very different, as well as the popularity of the alternative models at these time points. The same applies to other products that have relatively a short life span on
sales (from release to discontinued dates). In short, even if two users interact with the same item, if the two interactions occur at very different time points, the contexts for the two decision makings could be very different. The context here is reflected by the candidate items and their properties (e.g., their popularity ranking) at the “decision making” time.

Note that, there are two kinds of context changes, at user side and at system side, respectively. The context changes in our discussion refer to the context changes at the system side, more specifically, the changes that can be fully observed in an offline dataset. For example, new items are released from time to time, and outdated items are no longer receiving more interactions from any users after some time. Along time, different items accumulate different number of interactions at different time points.

Back to the illustration in Figure 3, if user $u_4$ purchases the same phone within the few days of $u_3$’s purchase, then likely the contexts for these two decision makings are very similar. Although it is hard to directly model the context changes between any two user-item interactions, it is reasonable to assume that if two interactions occur within a short time period, the context change at system side is not significant. The length of a reasonable “short time period” may vary from one system to another depending on the characteristics of the items (e.g., news, movie, music, book, restaurant, and consumer electronics) [30]. In other words, if a user has many interactions that occurred not too long ago from the test time, the contexts of the past interactions and the context of this test instance are similar. In this case, the test interaction may well align with the user preference learned from the past few interactions. This could be an explanation to why recommender models that are trained with recent parts of the dataset deliver better accuracy [30], and to why the users who have recent transactions enjoy better recommendations [13].

Because of the ignorance of timeline in our modeling, the possible context changes cannot be considered in mainstream RecSys models. Two user-item interactions that occurred 10 years apart are modeled in the same way as if the two interactions occurred within the same day.

If global timeline is so important, then why industry players are not highlighting this factor in modeling? One possible reason is that industry players often need to process a large volume of data. In their model evaluation (regardless of online or offline), the recent parts of data are sufficient for training. We can take some recently released datasets as examples. MIND dataset for news recommendation from Microsoft contains interactions of one million users randomly sampled in 6 weeks [32]. The user behavior data from Taobao for recommendation contains interactions of one million users randomly sampled in about one week. In terms of dataset size, these two datasets are not small; in terms of the time duration they cover, the system side contexts may not change much. In industrial-scale systems, Anil et al. [1] state that “limiting training data to more recent periods is intuitive.” The authors further comment that if the date range is extended further back in time, “the data becomes less relevant to future problems”. In industry setting, recommenders are often periodically retrained/updated with the recent data. In this sense, practical recommenders naturally consider data recency along timeline.

4 A FRESH LOOK AT RECSYS

In academic research, we often abstract similar real-world problems (e.g., the various types of recommendation problems in different domains) to a formal research problem. Accordingly, we propose evaluation metrics to quantify to what extent a proposed solution has addressed this

---

4 We do not consider context change at user side (e.g., job change or relocation to a different place), or changes that cannot be observed in a dataset like promotion campaign.

5 https://tianchi.aliyun.com/dataset/dataDetail?dataId=649&userId=1&lang=en-us

6 It is understandable that news recommendation is very sensitive to time due to the characteristics of the items, i.e., news.
problem on different datasets. Very often, a solution is designed largely for the purpose of achieving better evaluation scores.

Unfortunately, the abstraction process to reach a formal problem definition comes along with simplification of real-world problems. In the mainstream RecSys problem setting, the “time” factor has been largely ignored due to the simplification process. The current way of evaluation is also proposed based on the simplified research problem. To give a fresh look at the RecSys problem, the global timeline has to be part of the problem definition, to truly reflect the problem of learning from past interactions then to recommend unseen items. Accordingly, the evaluation methodology has to factor in the global timeline.

4.1 Meaningful and Practical Evaluation

We are not in short of papers on large-scale empirical evaluations [5, 25, 26, 34, 35]. However, the results reported in these papers may not be comparable to each other due to the ignorance of timeline. A recent benchmark [34] verifies that current evaluation methodology leads to recommending “future items” which would never occur in reality, a consistent finding earlier reported in [12]. Despite the existence of many large-scale empirical studies, there remain questions on reproducibility, and technical and theoretical flaws [4, 7]. On the other hand, it is a challenging problem to evaluate recommender systems, because the evaluation metrics can be defined from multiple perspectives [2, 33] and there remain many challenging issues even if these metrics are well defined [29], and even if the evaluation is conducted online [3].

We probably want to begin with something simple: Can we at least fairly compare our model with Popularity, the simplest baseline that has been used for decades? In other words, our recommenders shall be compared with a Popularity ranking that is used in practice, i.e., the ranking of items on a hourly, daily, or weekly basis.

In Figure 4, the vertical line in purple is an illustration of Popularity ranking of items at a time point slightly before the current time $t_c$. When users interact with the system at $t_c$, the popularity ranking provides the most popular items in the past $T_p$ time duration. $T_p$ can be one hour, one day, one month, or the entire history, i.e., a parameter depending on the characteristics of the items. This popularity method can be more precisely named as Recent Popularity where the recency $T_p$ is configurable. If $T_p$ is set to cover the entire duration of all existing training data, then the ranking is the most popular items in history. Note that the most popular in history remains different from the ill-defined Popularity baseline, because of the observation of the global timeline in Figure 4.

We now apply a similar concept to evaluate any RecSys model in offline setting. The first step in the evaluation is to split data into training and test sets. We may follow any existing data split scheme. In the following, the leave-one-out scheme is used as an example. With leave-one-out split, the last interaction of every user is in the test set and the remaining interactions of the user are in the training set. To observe the global timeline, all user-item interactions (in both train and
test) are arranged in chronological order by their timestamps. Then the entire timeline is split into time windows of size \( W \) as shown in Figure 5. We evaluate a model on all test instances within one time window \( W \) at each time, window by window. Suppose \( W_{n+2} \) is the current window for evaluation, within which \( t_{x1} \) and \( t_{x2} \) are the two test instances (see Figure 5). The model shall be able to learn from all or subset of the following data instances: (i) all training instances in \( W_{n+2} \), and (ii) all training and test instances in all the windows before \( W_{n+2} \). For each test instance, we compute the evaluation measures (e.g., precision/recall), and aggregate the results. The aggregation may happen at each window, or across all test instances in all windows.

We note the following points for the timeline evaluation scheme described above. First, this evaluation scheme observes global timeline by design. Second, this evaluation scheme still suffers from data leakage. However, the amount of data leakage is controlled by the size of time window \( W \). If the time window is reasonably small with respect to the entire timeline (e.g., one month vs ten years), then data leakage does not significantly affects the results. The amount of data leakage in each time window trades off the total number of windows in the entire evaluation. We may consider that the current mainstream offline evaluation is a special case of the timeline scheme: there is only one big time window covering the entire timeline duration. To completely avoid data leakage, the evaluation scheme can be changed to only allow the model to learn from previous windows when evaluating on the current time window. Then, we may have too few training instances for some users, like \( u_2 \) in Figure 5. Third, along timeline, the model needs to be retained or updated for evaluation on each time window. The design allows a model to learn from different sets of training instances. For example, recency popularity may derive the popularity ranking from only one or a few recent windows. Other machine learning based models may choose to use training instances from recent windows as well, i.e., only the recent parts of the data. In this sense, the number of recent windows/interactions becomes a parameter in model training. This is a different understanding of “fair comparison” from the current mainstream setting, where all models use the same set of training data.

The key consideration here is to observe global timeline in the evaluation process. The key assumption here is that the system context does not change much within a time window. As aforementioned, this evaluation scheme is not problem free, e.g., data leakage remains possible. However, exactly mimicking real-time setting would lead to a very complex evaluation process, which is less practical in academic research. Note that, the timeline scheme is just one of the possible ways to evaluate RecSys in a more practical manner and the scheme might have already been used in some previous studies. There are other data partition schemes that do not lead to data leakage, for example, partition by time point. Further, there is also a line of research in RecSys known as incremental learning. In this setting, a model learns from past data and predicts future interactions along timeline, by definition. The discussion here is to offer a revisit to the batch-oriented train/test offline evaluations that do not consider timeline.
4.2 Meaningful Modeling of User Preference

A recommender is expected to answer a user’s latent (information) needs. The user-item interactions are the observed results, or the answers to the earlier information needs. In the original design of collaborative filtering, users choose who to follow as the expected “information filters”. In the current RecSys model design, users are not empowered to proactively choose the information filters. Rather, the information filters are modeled based on users past interactions. However, such simplification in user preference modeling only captures the results of the decision makings, and does not capture the “contexts” of the decision makings. Even if two users interact with the same item, they may not make decisions in a similar context, particularly when the two decision makings occur a long time apart from each other. However, it is hard to model the decision context, given the limited data in academic research.

One possible way of evaluating similarity of decision contexts is through impressions. In simple words, impressions are a list of items presented to a user when she/he makes an interaction. For example, if user $u_1$ interacts with item $D$ when presented with impression $\{A, B, C, D\}$, and user $u_2$ also interacts with item $D$ with impression $\{D, E, F, G\}$, then the two decision makings are based on different contexts, although the final decisions are the same. Recently, a few datasets are made available with impressions, including ContentWise Impressions [22], MIND [32], and FINN.No Slates [6] datasets. Such datasets are of great value for exploring new ways of user preference modeling. There is also a study on evaluation of recommender systems with impressions [21].

Large-scale recommender systems typically consist of multiple steps like matching (i.e., candidates generation) and ranking [19, 35]. In the matching step, candidate items are identified from all available items. In the absence of impressions, the decision context might be reflected by these candidate items, provided that the matching step observes the global timeline and retrieves only the available items at the corresponding time point (i.e., a test instance’s timestamp). Here, the similarity of candidate items is a proxy to measure the similarity of two decision contexts. The reason of using candidate items instead of the final ranked items is that candidate items are less dependent on a particular ranking algorithm.

To generate candidate items for every interaction is very expensive. A simplification here could be an assumption that if two interactions happen within a very short time period, then the decision contexts are similar. Then, a function of time duration could be used to model the context changes between two interactions.

Modeling user preference with the consideration of decision context offers us a fresh look at many specific problems in RecSys. One of them is sequential recommendation. In sequential recommendation, interactions (or actions) of a user form a sequence and the task is to predict the user’s next interaction. A user sequence preserves the relative sequential order of her interactions, but does not record the timestamps of these interactions. Petrov and Macdonald [23] show that recent training interactions (in terms of sequential order) in a sequence better indicate the user’s interest. The recency by sequential order can be an approximation of the recency by timestamps. Nevertheless, without recording timestamps of the interactions in a sequence, it remains difficult to precisely model the context differences in decision making. For one user, her first and last interactions could be one year apart. For another user, all interactions in her sequence may occur within a week. The decision making contexts in these two sequences will be quite different. Hence, it would be more meaningful to model interactions with timestamps in sequences. By considering timestamps, we are also in a better position to evaluate whether some datasets are indeed suitable for sequential recommendation. For example, the MovieLens dataset is not suitable for sequential recommendation, because there is no meaningful sequence in the dataset [31].
5 CONCLUSION

We start with a few counter-intuitive observations made in recent studies; then we explain the reasons behind. One key reason is the ignorance of the global timeline in model evaluation, which even leads to a wrong implementation of the simplest baseline Popularity. Interestingly, because industry players often have to limit to recent data due to large data size, they may not highlight the importance of timeline. However, in academic research, many widely used datasets cover interactions recorded in a long period. Following a similar problem understanding as industry players but evaluating on datasets of different characteristics is the main contradiction here. The missing of timeline leads to improper evaluation of our models due to data leakage. Hence, the results reported in many papers become incomparable; the models developed in academic research are rarely transferable to practical systems. In this opinion paper, we highlight the importance of timeline in obtaining more meaningful evaluation results. On top of that, the consideration of timeline also provides us new insights in modeling user preference. We shall not only focus on the user-item interactions, which are the results of decision makings, but also focus on the contexts of decision makings. After all, we aim to model user preference in making decisions.

REFERENCES

[1] Rohan Anil, Sandra Gadanho, Da Huang, Nijith Jacob, Zhuoshu Li, Dong Lin, Todd Phillips, Cristina Pop, Kevin Regan, Gil I. Shamir, Rakesh Shivanna, and Qiqi Yan. 2022. On the Factory Floor: ML Engineering for Industrial-Scale Ads Recommendation Models. In Workshop on Online Recommender Systems and User Modeling (ORSUM) at RecSys22. https://doi.org/10.48550/ARXIV.2209.05310

[2] Pablo Castells and Alistair Moffat. 2022. Offline recommender system evaluation: Challenges and new directions. AI Magazine 43, 2 (2022), 225–238. https://doi.org/10.1002/aaai.12051

[3] Hung-Hsuan Chen, Chu-An Chung, Hsin-Chien Huang, and Wen Tsui. 2017. Common Pitfalls in Training and Evaluating Recommender Systems. SIGKDD Explorations 19, 1 (2017), 37–45.

[4] Paolo Cremonesi and Dietmar Jannach. 2021. Progress in Recommender Systems Research: Crisis? What Crisis? AI Magazine 42, 3 (Nov. 2021), 43–54. https://doi.org/10.1009/aimag.v42i3.18145

[5] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In RecSys. ACM, 101–109.

[6] Simen Eide, David S Leslie, Arnoldo Frigessi, Joakim Rishaug, Helge Jenssen, and Sofie Verrewaere. 2021. FINN.No Slates Dataset: A New Sequential Dataset Logging Interactions, All Viewed Items and Click Responses/No-Click for Recommender Systems Research. In RecSys. 556–558. https://doi.org/10.1145/3460231.3474607

[7] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. 2021. A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research. ACM Trans. Inf. Syst. 39, 2, Article 20 (jan 2021), 49 pages. https://doi.org/10.1145/3434185

[8] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. 1992. Using Collaborative Filtering to Weave an Information Tapestry. Commun. ACM 35, 12 (dec 1992), 61–70. https://doi.org/10.1145/138859.138867

[9] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In SIGIR. ACM, 639–648.

[10] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In WWW. ACM, 173–182.

[11] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2020. A Re-visit of the Popularity Baseline in Recommender Systems. In SIGIR. ACM, 1749–1752.

[12] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2022. A Critical Study on Data Leakage in Recommender System Offline Evaluation. CoRR abs/2010.11060 (2022).

[13] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2022. Do Loyal Users Enjoy Better Recommendations? Understanding Recommender Accuracy from a Time Perspective. In ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR) (Madrid, Spain). ACM, 92–97. https://doi.org/10.1145/3539813.3545124

[14] Wang-Cheng Kang and Julian J. McAuley. 2018. Self-Attentive Sequential Recommendation. In ICDM. ACM, 197–206.

[15] Sergey Kolesnikov and Mikhail Andronov. 2022. CVTT: Cross-Validation Through Time. CoRR abs/2205.05393 (2022).

[16] Sara Latifi and Dietmar Jannach. 2022. Streaming Session-Based Recommendation: When Graph Neural Networks Meet the Neighborhood. In RecSys (Seattle, WA, USA). ACM, 420–426. https://doi.org/10.1145/3523227.3548485
[17] Jiacheng Li, Yujie Wang, and Julian J. McAuley. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. In WSDM. ACM, 322–330.

[18] Yangkun Li, Mohamed-Laid Hedia, Weizhi Ma, Hongyu Lu, Min Zhang, Yiqun Liu, and Shaoping Ma. 2022. Contextualized Fairness for Recommender Systems in Premium Scenarios. Big Data Research 27 (2022), 100300. https://doi.org/10.1016/j.bdr.2021.100300

[19] Weiwen Liu, Yunxia Ji, Jiarui Qin, Fei Sun, Weinan Zhang, Rui Zhang, and Ruiming Tang. 2022. Neural Re-ranking in Multi-stage Recommender Systems: A Review. In IJCAI, Luc De Raedt (Ed.). ijcai.org, 5512–5520. https://doi.org/10.24963/ijcai.2022/771

[20] Zaiqiao Meng, Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2020. Exploring Data Splitting Strategies for the Evaluation of Recommendation Models. In RecSys. ACM, 681–686.

[21] Fernando Benjamin Perez Maurera, Maurizio Ferrari Dacrema, and Paolo Cremonesi. 2022. Towards the Evaluation of Recommender Systems with Impressions. In RecSys (Seattle, WA, USA). ACM, 610–615. https://doi.org/10.1145/3523227.3551483

[22] Fernando B. Pérez Maurera, Maurizio Ferrari Dacrema, Lorenzo Saule, Mario Scriminaci, and Paolo Cremonesi. 2020. ContentWise Impressions: An Industrial Dataset with Impressions Included. In CIKM (Virtual Event, Ireland). ACM, 3093–3100. https://doi.org/10.1145/3340531.3412774

[23] Aleksandr Petrov and Craig Macdonald. 2022. Effective and Efficient Training for Sequential Recommendation Using Recency Sampling. In RecSys (Seattle, WA, USA). ACM, 81–91. https://doi.org/10.1145/3523227.3546785

[24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In UAI. AUAI Press, 452–461.

[25] Zhu Sun, Hui Fang, Jie Yang, Xinghua Qu, Hongyang Liu, Di Yu, Yew-Soon Ong, and Jie Zhang. 2022. DaisyRec 2.0: Benchmarking Recommendation for Rigorous Evaluation. CoRR abs/2206.10848 (2022). https://doi.org/10.48550/arXiv.2206.10848

[26] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. 2020. Are We Evaluating Rigorously? Benchmarking Recommendation for Reproducible Evaluation and Fair Comparison. In RecSys. ACM, 23–32.

[27] Sylwia Sysko-Romańczuk, Piotr Zaborek, Anna Wróblewska, Jacek Dąbrowski, and Sergiy Tkachuk. 2022. Data modalities, consumer attributes and recommendation performance in the fashion industry. Electronic Markets (Aug 2022). https://doi.org/10.1007/s12525-022-00579-3

[28] Pablo Sánchez and Alejandro Bellogín. 2021. On the effects of aggregation strategies for different groups of users in venue recommendation. Information Processing & Management 58, 5 (2021), 102609. https://doi.org/10.1016/j.ipm.2021.102609

[29] Yan-Martin Tamm, Rinchin Damdinov, and Alexey Vasilev. 2021. Quality Metrics in Recommender Systems: Do We Calculate Metrics Consistently?. In RecSys (Amsterdam, Netherlands). ACM, 708–713. https://doi.org/10.1145/3460231.3478848

[30] Robin Verachtert, Lien Michiels, and Bart Goethals. 2022. Are We Forgetting Something? Correctly Evaluate a Recommender System With an Optimal Training Window. In Perspectives on the Evaluation of Recommender Systems Workshop (PERSPECTIVES) at RecSys22. Seattle, WA, USA.

[31] Daniel Woolridge, Sean Wilner, and Madeleine Glick. 2021. Sequence or Pseudo-Sequence? An Analysis of Sequential Recommendation Datasets. In Proceedings of the Perspectives on the Evaluation of Recommender Systems Workshop co-located RecSys (CEUR Workshop Proceedings, Vol. 2955), Eva Zangerle, Christine Bauer, and Alan Said (Eds.). CEUR-WS.org.

[32] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, and Ming Zhou. 2020. MIND: A Large-scale Dataset for News Recommendation. In ACL. Association for Computational Linguistics, Online, 3597–3606. https://doi.org/10.18653/v1/2020.acl-main.331

[33] Eva Zangerle and Christine Bauer. 2022. Evaluating Recommender Systems: Survey and Framework. ACM Comput. Surv. (jul 2022). https://doi.org/10.1145/3556536

[34] Wayne Xin Zhao, Zihan Lin, Zhichao Feng, Pengfei Wang, and Ji-Rong Wen. 2022. A Revisiting Study of Appropriate Offline Evaluation for Top-N Recommendation Algorithms. ACM Trans. Inf. Syst. (may 2022).

[35] Jieming Zhu, Quanyu Dai, Liangcai Su, Rong Ma, Jinyang Liu, Guohao Cai, Xi Xiao, and Rui Zhang. 2022. BARS: Towards Open Benchmarking for Recommender Systems. In SIGIR (Madrid, Spain). ACM, 2912–2923. https://doi.org/10.1145/3477495.3531723