

Modeling Online Behavior in Recommender Systems: The Importance of Temporal Context

Milena Filipovic^{1,2}, Blagoj Mitrevski^{2,3}, Diego Antognini², Emma Lejal Glaude¹, Boi Faltings² and Claudiu Musat¹

¹Swisscom, Switzerland

²Ecole Polytechnique Fédérale de Lausanne, Switzerland

³Symphony, North Macedonia

Abstract

Recommender systems research tends to evaluate model performance offline and on randomly sampled targets, yet the same systems are later used to predict user behavior sequentially from a fixed point in time. Simulating online recommender system performance is notoriously difficult and the discrepancy between online and offline behaviors is typically not accounted for in offline evaluations. This disparity permits weaknesses to go unnoticed until the model is deployed in a production setting. In this paper, we first demonstrate how omitting temporal context when evaluating recommender system performance leads to false confidence. To overcome this, we postulate that offline evaluation protocols can only model real-life use-cases if they account for temporal context. Next, we propose a training procedure to further embed the temporal context in existing models. We use a multi-objective approach to introduce temporal context into traditionally time-unaware recommender systems and confirm its advantage via the proposed evaluation protocol. Finally, we validate that the Pareto Fronts obtained with the added objective dominate those produced by state-of-the-art models that are only optimized for accuracy on three real-world publicly available datasets. The results show that including our temporal objective can improve recall@20 by up to 20%

Keywords

evaluation, recommendation, offline and online evaluation, multi-objective optimization

1. Introduction


In an increasingly digital world, recommender systems are a staple of our daily routines. They influence how we perceive our environment, from media content to human relationships. Traditional methods of evaluation that entail random sampling over a long period of time are perfect for a system that is designed to remain unchanged for an equally long and predefined period. However, if the system is to be used in a dynamic setting, e.g. recommending the next song to play in a playlist, the way it is evaluated must reflect that. Inadequate evaluation techniques can lead to false confidence, which is especially detrimental in commercial settings.

Perspectives on the Evaluation of Recommender Systems Workshop (PERSPECTIVES 2021), September 25th, 2021, co-located with the 15th ACM Conference on Recommender Systems, Amsterdam, The Netherlands

✉ milena.filipovic1@swisscom.com (M. Filipovic); blagoj.mitrevski@symphony.is (B. Mitrevski); diego.antognini@epfl.com (D. Antognini); emma.lejalglaude@swisscom.com (E. L. Glaude); boi.faltings@epfl.com (B. Faltings); claudiu.musat@swisscom.com (C. Musat)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

Recommender system evaluation can be done *online* or *offline*. *Online* evaluation implies deployment of the recommender in the real world, often in a commercial setting. While this may be ideal for measuring the real-life impact of a system, it is also costly, both in terms of resources and time, and therefore rarely used in research and benchmarking. In *offline* evaluation, historical data is utilized. Some portion of the data is selected to train on, while another subset is used for performance testing. Since all data points are available beforehand, the evaluation costs and overall timeline are significantly reduced.

Irrespective of whether they are evaluated online or offline, many existing recommenders ignore temporal information. Most recommender systems fall into one of two main categories: content-based or collaborative filtering [1, 2, 3, 4]. The former relies on recommended items having similar attributes to those that the user has previously interacted with, while the latter methods base their recommendation on items bought by similar users. However, many models completely ignore temporal information, with the notable exception of time-aware recommender systems[5]. These systems introduce additional context to interactions: their temporal dimension.

In this work, we first focus on the importance of temporal dynamics in recommender system evaluation. We draw attention to the lack of standardization in the evaluations, and the differences between research settings and the systems' ultimate applications. Then, we highlight two temporal evaluation protocols and show how they attain a closer approximation of the real-life conditions in which recommender systems are deployed. Second, we present a multi-objective approach[6] of incorporating the temporal context to time-unaware recommender systems without any change in model architecture. We introduce a naive recency objective as a means to include temporal dynamics in typically time-independent recommender systems. We also provide a measure of recency in the form of a performance metric. Through experiments on three real-world publicly available datasets we show that the addition of the naive temporal objective yields improvements not only in recency but also in relevance. Finally, we demonstrate that the Pareto Fronts obtained with the added objective dominate those produced by state-of-the-art models.

To the best of our knowledge, this is the first study quantifying the difference in recommender system performance when evaluated using methods that model real-world environments, as opposed to traditional techniques. We also show that a recommender system can be optimized for both relevance and recency objectives simultaneously. To summarize, the main contributions of this paper are as follows:

- We demonstrate how commonly used evaluation protocols do not provide adequate modeling of real-world deployment settings. To combat this, we show two evaluation techniques to facilitate offline modeling of online production environments that inherently incorporate temporal dynamics;
- We introduce a “naive” recency function that can be utilized to create a temporal objective. We show that optimizing for both temporal context and relevance [6] leads to solutions that dominate those optimized just for relevance in both dimensions.

2. Related Work

2.1. Evaluating Recommender Systems

2.1.1. Traditional Recommender Systems.

Inputs and outputs share similarities with classification and regression modeling: a class variable is predicted from a set of given features. Given that recommendation tasks can be seen as a generalization of these, some evaluation techniques used for classification are transferable to recommender systems.

In collaborative filtering research, recommenders are generally evaluated either through *strong* or *weak generalization*, as characterized by [7]. In both approaches, models are trained on observed interactions and validated or tested on those that are held-out. However, there exist some key differences. *Weak generalization* is introduced in [8], where the held-out set is created through random sampling of the available interactions of all users. *Strong generalization* differs by taking disjoint sets of users for the training, validation, and testing sets. Following this, some interactions are held-out from the validation and test sets and then approximated using the recommender. Methods that encode user representation cannot apply *strong generalization*, as they cannot generate outputs for previously unseen users. An example of the *strong generalization* approach can be seen in [3], whereas [9, 10, 11] all use *weak generalization*.

Several of these works emphasize that the application of their recommender system would be in predicting future user actions, yet all validation and testing is done with randomly selected interactions. This can break the time linearity as the knowledge of future interactions during training can help predict a randomly sampled past interaction. While much effort is directed towards establishing the importance of proper evaluation design, it is generally focused on implementing relevant metrics to avoid under- or over-estimating real-world performance [12], and not on the evaluation procedures themselves.

2.1.2. Temporal Recommender Systems.

They denote time-aware RS (TARS), and incorporate time explicitly or implicitly[5].

Temporal recommender systems can be taken to include sequence-aware recommender systems (SARS), as a special form of time adaptive recommenders that focus on ordering rather than specific time instances[5]. It is however important to note that while they can be evaluated using similar techniques, SARS approach temporal dynamics from a different perspective, therefore the resulting models can differ greatly from typical TARS[13]. [5] provide an extensive overview of possible evaluation techniques, which served as an inspiration and point of reference for this work. While traditional evaluation protocols may be used on temporal recommenders, it is more representative to preserve the temporal ordering between interactions since this is something that the recommender aims to learn. By extension, train, validation, and test splits should also be ordered. [13] state that they were unable to find a consensus among evaluation protocols used in recent sequence-aware recommender work, which is mirrored in our findings. Yet we did determine that most recent SARS focus only on next item prediction, meaning they output one recommendation. They also typically employ certain target item conditions to decrease computational cost [5]. The target item conditions determine the (sub)set of items for

which a recommender should produce predictions and are specific for top-N recommendation evaluation. The reduction of the computational costs is generally done through conditions that rank one ground truth item against a set of other items false items. Examples can be found in [14, 15, 16]. We return to the problem of subsampling in Section 3.

2.2. Temporal Context in Recommender Systems

In this paper, we introduce the concept of recency. An important note is that there are multiple definitions of recency in recommender systems literature. In fact, this lack of consensus has persisted for years. [17, 18] treat the recency of an item as an attribute that is user-dependent. The value is determined by the last time the user interacted with a given item. [19, 20] also claim to incorporate recency into their research: when recommending news articles, they measure recency as the age of the item on the platform. Our analysis will follow the latter definition. This is in line with our desire to explore the effects of a light-weight temporal addition on the performance of traditional RS. Further work to determine the "ideal" definition of recency, while undoubtedly invaluable, is outside the scope of this work.

3. Evaluation Protocols

We propose that the temporal dimension should be considered when evaluating the performance of any recommender. While random sampling may be an appropriate target selection technique for some classification or regression tasks, we argue that this is not the case when it comes to predicting a user's subsequent move.

Unlike the vast majority of evaluation methods applied to traditional recommenders, temporal recommender systems literature does model the passage of time. However, as stated above, the performance is often computed over a subset of the itemset and the user's true chosen item. The argument is that subsampling is necessary due to the complexity of the ranking task. While this has some validity, itemsets of around 10,000 datapoints can be ranked highly efficiently, especially when taking into consideration recent advancements in machine learning libraries and GPU programming. Therefore, we do not utilize subsampling in our work.

The adoption of a recommender system in real scenarios has two major phases. The first, called the development phase, is purely offline and theoretical. In this part, three separate sets of data must be created: a training set that the model will use to learn item and user representations, a validation set for hyperparameter tuning, and a test set to evaluate how well the model performs. The second, called the deployment-ready phase, include interactions with end-users. The maximum amount of data is leveraged to train a model with as much information as possible, evaluate its performance, and then deploy it into production. In this case, only two sets are needed: training and validation sets.

One downside of collaborative filtering methods is that most models are incapable of incorporating new items without retraining. While ways to alleviate this problem have been explored [21], the issue remains widespread and worthy of more study, but lies outside the scope of this paper. Therefore, we assume an industry-like environment: the recommender system will be retrained regularly and will be exposed to clients for a relatively short period, ranging from a

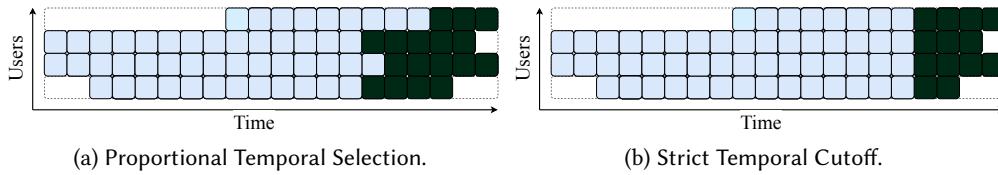


Figure 1: Two methods for temporal validation set target selection.

couple of days to a few months. We postulate that the performance of the recommender on the last portion of historically available data is most indicative of how it will behave when deployed.

Our protocols focus on set creation. When selecting the target values in a validation set, we take two possible approaches. The first, *proportional selection*, depicted in Figure 1a, selects the final $X\%$ of each user’s interactions and uses these to create target items. Here we preserve the time ordering of the input and target interactions, maintaining similarity with the real-life use-case. The second approach, shown in Figure 1b, is based on a *strict time cutoff* to select the target items of the validation set. This method is even closer to the real-world use case. However, it does suffer from certain drawbacks as user interactions are not necessarily evenly distributed through time, leading to some users being more represented than others in the target set. While these are similar to the suggestions developed in [5], we underline that these approaches should not be limited to evaluating TARS. It is crucial to approximate with maximum precision the performance of a model when developing a novel system, before it is released into production. The second approach directly models the real-world context and contains user-item interaction sequences created after a specific strict time cutoff.

4. Recency to Improve Recommendation

The main task of a recommender system is to anticipate users’ future desires and suggest relevant content. The *relevance* objective is the one that is most commonly found in recommender systems literature and accounts for the accuracy or correctness. It actively focuses the recommender on selecting the item(s) with which the users will most likely interact.

However, just recommending the most relevant items does not always satisfy all the concerns of those building the system and it is not the only objective used in practice. We distinguish two types of objectives: *correlated* and *uncorrelated* to relevance. The former ones correspond to those whose optimization is linked to the relevance objective. Examples are novelty [22], serendipity [23], and utility-based objectives, such as revenue. For the latter, *not correlated to relevance*, examples can be found in diversity and fairness. In this work, we introduce a simple utility-based objective used to inject temporal information alongside the relevance objective. While the exploration of uncorrelated objectives is essential for the future of recommender systems, we leave it for future work.

4.1. Adding Temporal Context

Based on our experience with real-life use-cases, we observed that users seem to gravitate towards content that had more recently been added to a given platform. While we cannot disclose internal facts and figures, the temporal objective described below was motivated by behaviour exhibited across many months of user interactions observed internally. Building on these findings, and works such as [19] and [20], we chose to explore the effects of incorporating recency as an objective during the learning phase. Given an item x with a timestamp t_x , we further define the recency function $f(\cdot)$ as:

$$f(x) = \begin{cases} 1 & \frac{t_x - t_{min}}{t_{max} - t_{min}} \geq 0.8 \\ 0.3^{(0.8 - \frac{t_x - t_{min}}{t_{max} - t_{min}}) \times \frac{10}{3}} & otherwise \end{cases} \quad (1)$$

where t_{max} and t_{min} are the maximum (most recent) and minimum (oldest) timestamps over the itemset. In $f(\cdot)$, we first scale all timestamps to $[0, 1]$ using the min-max scaler, and then apply a transformation inspired by [24].

We underline that this is a naive function which we found to best approximate user interactions observed in-house. Works such as [25] rely on a power law distribution to model temporal effects, highlighting that further exploration into use-case specific temporal context approximations may yield exciting results.

The recency objective is formulated as a loss that stimulates the recommendation of recent items. Each item in the itemset is assigned a recency weight, The vector is then used to weigh item importance when calculating the loss. Adding weights into a traditional loss does not affect the differentiability of the function.

To illustrate how our temporal objective can be easily integrated into a traditionally time-unaware RS, we take as a use-case the state-of-the-art variational autoencoder Mult-VAE^{PR} of [3]. For the sake of brevity, we refer the reader to [3] for more details about the model. We thus propose an extension of Mult-VAE^{PR}, where the loss function for user u is modified to:

$$\mathcal{L}_\beta(x_u; \theta, \phi) = \mathbb{E}_{q_\phi(z_u|x_u)} [\log p_\theta(f(x_u) * x_u|z_u)] - \beta \cdot \text{KL}(q_\phi(z_u|x_u)||p(z_u)) \quad (2)$$

where the expected negative log-likelihood is modified to include the element-wise multiplication of input vector x_u by $f(x_u)$, which corresponds to the recency scores of the given items in x_u . β controls how much importance is given to the KL term, z_u is a variational parameter of the variational distribution θ and ϕ are model parameters.

4.2. Multi-Objective Optimization

Optimizing a recommender on multiple objectives is non-trivial. Thanks to the recent work of [6], we employ the proposed multi-gradient descent algorithm for multiple objectives to train our recommenders. After a standard forward pass, the loss and gradient are computed for each objective and weights of the gradients are computing as a Quadratic Constrained Optimization Problem [26]. This can be solved analytically for two objectives, or solved as a constrained optimization problem as proposed in [27] for more than two objectives. Solving it allows us to obtain the common descent vector and update the parameters. This training procedure enables

us to incorporate both our temporal context and the relevance objectives to retrieve time-aware recommendations. The algorithm adapts the weight repartition between the two objectives in an advanced manner to optimize both during training.

5. Experiments

5.1. Datasets

We study the performance of various models on three real-world publicly available datasets.

MovieLens-20M. Contains about 20 million ratings¹, with values between 1 and 5. We binarize the user-item interaction matrix, keeping ratings of 4 and above as positive feedback to transform it into implicit feedback. We filter out all users with less than five ratings, and all movies rated by less than five users. We focus on the last ten years of available data (2005-2015). The preprocessed dataset contains 46,295 users, 9479 items, and 3.76M interactions with a density of 0.86%.

Steam. Has review information from the gaming platform Steam². We converted user-item interactions into a positive feedback signals. The dataset contains reviews from 2010 to 2018; however, the platform only sees an uptick in review activity after 2014, therefore we use 2014-2018 for our analysis. After preprocessing the dataset contains 471,457 users, 13,018 items, and 3.14M interactions with a density of 0.09%.

Netflix. The well-known Netflix Prize Competition dataset³. It consists of over 100 million ratings. We filter these ratings in the same way as the MovieLens ratings, and take the last two years of activity (2003-2005). Because of low performance on certain baselines, we denote two variants for the implicit feedback: one with threshold of 4 and above ($\text{Netflix} \geq 4$), the other one with a threshold of 5 ($\text{Netflix} \geq 5$). After preprocessing the dataset contains 257,775 users, 13,995 items, and 38.87M interactions with a density of 0.59%.

5.2. Recommendation Techniques

We conduct experiment with the following well-known recommendation systems:

Mult-VAE^{PR}. The MAMO framework⁴ and the setup from the original paper [3] are utilized.

SVD. The PyTorch implementation⁵ of the Singular Value Decomposition [28] is used, taking the top 100 dimensions.

¹<https://grouplens.org/datasets/movielens/20m/>.

²<https://cseweb.ucsd.edu/~jmcauley/datasets.html>.

³<https://www.kaggle.com/netflix-inc/netflix-prize-data>.

⁴<https://github.com/swisscom/ai-research-mamo-framework>.

⁵<https://pytorch.org/docs/stable/generated/torch.svd.html>.

NCF. The Neural Collaborative Filtering [2], we take the implementation from ⁶, sample 4 negative instances for every existing user-item interaction, set the predictive factor of 64, and the number of hidden layers for the multilayer perceptron (MLP) to three. We do not present results obtained using pre-trained NeuMF, as they exhibited the same patterns as generalized matrix factorization (GMF) and MLP, but did not give a significant improvement. To resolve difficulties in obtaining good results with *Netflix*_{≥4} for GMF and MLP models, we used instead the *Netflix*_{≥5} dataset.

BERT4Rec. We implement this sequence-aware recommender system from [14] in PyTorch and integrate it with the MAMO framework. We take this model to show how directly encoding temporal information in the model impacts performance. In this case the ordering represents the temporal information. Hyperparameters were mostly taken from the original paper, otherwise selected based on a simple grid search. The number of transformer layers is set to 2, the head number is 4, head dimensionality is 64, and the dropout is 0.1. We use a sequence length of 100, while the proportion of masked inputs is 0.2. The model is trained using the Adam optimizer with a learning rate of 1e-4. All models were trained with the Adam optimizer, with a learning rate of 0.001.

5.3. Experimental Setup

We explore whether validation set formation in the *deployment-ready* phase may lead to false confidence in the performance of the evaluated model. In the *deployment-ready* phase, what we call the validation set is not necessarily used for hyperparameter tuning, but to assess the performance of the model before it is deployed. There are minor differences in the datasets used for models with and without user representation. Models without user representation require some input interactions to be able to predict targets, while those with simply need to be passed a user identifier. We divide our experiments into three sets, corresponding to the type of evaluation.

5.3.1. Traditional Evaluation.

Similarly to [3], we divide the users 80:10:10 to form a train, validation, and test set. The target interactions are selected by randomly sampling 20% of the user-item interactions in the validation and test sets.

We show that if a model is evaluated on and then applied to a task that entails predicting randomly held-out interactions, the performance achieved on both validation and test sets is comparable. This traditional approach is typically used to report model performance.

We then contrast performance on randomly held-out interactions in the validation set against temporally held-out interactions in the test set. We take 5% of the users from the train set to create the validation set and randomly hold-out 20% of their interactions. The train and validation sets contain user-item interactions up to a specific point in time. The test set contains the interactions and users from the train and validation sets as inputs, and the temporally held-out interactions are targets, to simulate deployment in a commercial setting.

⁶<https://github.com/guoyang9/NCF>.

5.3.2. Temporal Evaluation.

We show that when evaluated with either a proportional or hard temporal cutoff, the model’s performance is closer to what would be observed in a real-life setting. However, it is important to note the ideal evaluation technique is heavily domain dependent.

We divide the train, validation, and test sets as follows. 5% of the users from the train form the validation set. In the first approach, we hold out the last 20% of user-item interactions from each user in the validation set. While in the other, we hold out the last couple of months of activity and evaluate the model’s ability to predict these interactions. The test set contains the interactions and users from the two other sets as inputs, and the temporally held-out interactions are targets.

5.3.3. Temporal Evaluation with Added Temporal Context.

We introduce temporal context into the traditionally time-independent Mult-VAE^{PR} by using the work from [6] to optimize the model for accuracy and recency. To calculate the recency score we take the timestamp of the moment that the item first became available, or the first recorded instance of any user interacting with the given item. This timestamp is t_x in the recency function 1. The strict temporal cutoff validation set is utilized, as well as the temporal test set described previously.

5.4. Evaluation Metrics

We evaluate models using three ranking metrics, as RS can often only show a predefined number of recommendations. We ensured that the items that the user had previously interacted with were removed from the output before the top-k results were selected for metric calculation.

- *Precision@K*: calculates how many of the recommended items are relevant to the user;
- *Recall@K*: quantifies the proportion of relevant items in the top-k recommended items by calculating how many of the desirable items are suggested to the end-user. We take our definition from [3];
- *Recency@K*: assigns a recency score to each item, calculating the rating of the top-k recommended and relevant items. For user u with relevant items I_u we define $\omega(k)$ as the item at rank k , where \mathbb{I} is the indicator function:

$$Recency@K(u, \omega, f) = \sum_{k=1}^K \mathbb{I}[\omega(k) \in I_u] \times f(\omega(k)) \quad (3)$$

6. Results

6.1. Traditional Evaluation.

This experiment aims to show that the traditional way of evaluation recommender systems, shown in Table 1, is not a faithful representation of the environments in which they are actually

Table 1

Results of initial Mult-VAE^{PR} experiments, evaluated on a traditional evaluation protocol. We report Recall / Precision at $k = 20$.

Dataset	Val ^{trad}	Test ^{trad}
ML-20M	0.31 / 0.17	0.31 / 0.17
Steam	0.20 / 0.02	0.20 / 0.02
Netflix _{≥4}	0.35 / 0.19	0.35 / 0.19

Table 2

Results of the Mult-VAE^{PR}, SVD, GMF, and MLP evaluated on a traditional, proportionally selected temporal, and strict cutoff validation set, as well as on a temporally held out test set. Results of BERT4Rec evaluated on a strict cutoff validation set and a time delayed test set. We report Recall / Precision at $k = 20$.

Dataset	Model	Val ^{trad}	Val ^{prop}	Val ^{cutoff}	Test ^{temp}
ML-20M	Mult-VAE ^{PR}	0.32 / 0.18	0.26 / 0.13	0.11 / 0.06	0.11 / 0.07
	SVD	0.25 / 0.22	0.14 / 0.11	0.07 / 0.03	0.11 / 0.07
	GMF	0.25 / 0.22	0.11 / 0.10	0.08 / 0.03	0.10 / 0.07
	MLP	0.25 / 0.23	0.12 / 0.10	0.07 / 0.03	0.11 / 0.07
	BERT4Rec	-	-	0.20 / 0.09	0.15 / 0.08
Steam	Mult-VAE ^{PR}	0.20 / 0.02	0.14 / 0.02	0.11 / 0.01	0.13 / 0.01
	SVD	0.10 / 0.02	0.10 / 0.02	0.09 / 0.01	0.08 / 0.01
	BERT4Rec	-	-	0.21 / 0.02	0.17 / 0.02
Netflix _{≥4}	Mult-VAE ^{PR}	0.35 / 0.18	0.22 / 0.10	0.12 / 0.05	0.10 / 0.05
	SVD	0.23 / 0.16	0.23 / 0.16	0.09 / 0.05	0.07 / 0.04
	BERT4Rec	-	-	0.24 / 0.13	0.20 / 0.05
Netflix _{≥5}	SVD	0.23 / 0.10	0.23 / 0.11	0.12 / 0.05	0.09 / 0.03
	GMF	0.31 / 0.14	0.30 / 0.14	0.14 / 0.05	0.12 / 0.04
	MLP	0.31 / 0.14	0.30 / 0.14	0.14 / 0.05	0.12 / 0.04

deployed. The good performance achieved by evaluating in this way can provide a false sense of security.

Our claim is supported by the values highlighted by Table 2. Even though the validation sets are not identical to the ones before, the performance observed is very similar. However, it degrades on the time delayed test set, or to be more precise, when we simulate what would happen in a production setting. Drops in performance of -65.63%, -35.00%, and -71.43% can be observed, on the Recall@20 values. We postulate that this discrepancy leads to significant dissonance between the results of certain recommenders as reported in literature, and those observed in their real-life application.

6.2. Temporal Evaluation.

The results shown in Table 2 depict what happens when using traditional validation as opposed to our proposed evaluation sets. The table illustrates how the strict cutoff validation set

approximates the *deployment* behavior. For all datasets, this approach seems to be a closer estimation of the "real-life" performance. For example, the drop in performance is reduced from -71.43% to -16.67% on the $\text{Netflix}_{\geq 4}$ dataset for the $\text{Mult-VAE}^{\text{PR}}$ model. The proportionally selected validation sets seems to work well for the Steam dataset, and we know from industry experience that it can be good on others. However, this seems to be highly dataset specific.

Table 2 also shows that this phenomenon is not isolated to the $\text{Mult-VAE}^{\text{PR}}$, but can be repeated with the SVD, GMF, and MLP models. As mentioned before, we were unable to conduct experiments on $\text{Netflix}_{\geq 4}$ with the GMF and MLP models; therefore we report their results on $\text{Netflix}_{\geq 5}$. It is important to note that simpler methods, especially those based on matrix factorization, do not deal well with the Steam dataset. This is the sparsest dataset that we work with which seems to make it difficult to learn anything meaningful. Based on this, we exclude the Steam dataset for GMF and MLP. However, we keep the results for SVD.

We strongly recommend that these evaluation methods be taken into account when presenting novel achievements in the field. When feasible, we recommend to apply both protocols.

6.3. Temporal Evaluation and Temporal Models.

The results presented so far were achieved using traditional recommender architectures, with no way of learning temporal dynamics. To show that it is possible to achieve better results on the given datasets, we incorporate the temporal dynamics into the training process, by utilizing BERT4Rec. The results are shown in 2, and dominate all traditional solutions. This confirms our hypothesis that temporal dynamics should be accounted for in both evaluation design and model architecture in order to attain the best possible recommenders.

With BERT4Rec the interaction ordering is encoded in the model. The authors acknowledge that the naive recency objective does not reproduce this effect when added to traditional RS. However, the goal of the subsequent subsection is to illustrate that the simple addition of a cheap time-dependant weight affects performance in a meaningful way.

6.4. Temporal Evaluation with Added Temporal Context.

To integrate the temporal context into the traditional models, our following contribution has the recency included as an objective influencing the optimization. We refer to the multi-objective $\text{Mult-VAE}^{\text{PR}}$ as the Multi-Objective Recency Enriched $\text{mult-VAE}^{\text{PR}}$ (MOREVAE).

We present both the Pareto Fronts obtained during training and the results of the best models on the test sets. These results were obtained through more intense training than those shown in the previous sections in an attempt to extract the best possible performance from the $\text{Mult-VAE}^{\text{PR}}$. The Pareto Fronts were generated by evaluating on the strict cutoff validation sets during training, and the best models were chosen by selecting those with the highest Recall@20 and applying them to the time delayed test sets. Figure 2 shows that the multi-objective approach not only dominates the single objective one in terms of recency, but that optimizing for recency also increases the relevance of the recommendations, validating our initial intuition. The results of the best models over the test sets are shown in Table 3. The improvements obtained are 18.18%, 0.00%, and 20% for Recall@20 ; 14.29%, 0.00%, and 25.00% for Precision@20 . The improvements seen in Recency@20 are 104.35%, 20.00%, and 94.12%.

Table 3

Comparison of Mult-VAE^{PR} and MOREVAE results obtained on temporally held out test sets. We report Recall, Precision, and Recency at $k = 20$.

Dataset	Model	R	P	Re
ML-20M	Mult-VAE ^{PR}	0.11	0.07	0.23
	MOREVAE	0.13	0.08	0.47
Steam	Mult-VAE ^{PR}	0.13	0.01	0.15
	MOREVAE	0.13	0.01	0.18
Netflix _{≥4}	Mult-VAE ^{PR}	0.10	0.04	0.34
	MOREVAE	0.12	0.05	0.66

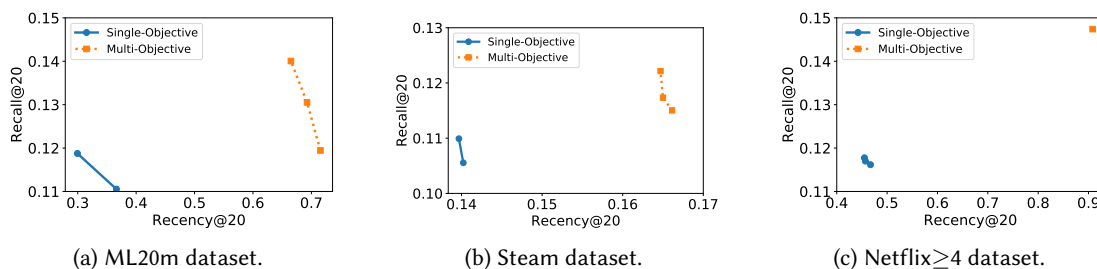


Figure 2: Pareto Fronts obtained through optimizing on one objective (accuracy), and two objectives (accuracy and recency).

7. Conclusion

Following standard offline recommendation evaluations during development may lead to false confidence when deploying models in real-life scenarios. Utilizing random sampling to hold out data is not an adequate approximation of many real-life use-cases. Previous research generally focuses on developing better metrics to reflect real-world performance, but still omits temporal context. We highlight this lack of standardization and propose two temporal evaluation protocols that empirically better approximate real-life conditions.

Our second contribution is to propose leveraging a multi-objective approach and train models on relevance and recency simultaneously. We show that a naive recency objective can be used to integrate temporal information in existing time-unaware recommenders. Experiments on three real-world publicly available datasets demonstrate that our method produced solutions that strictly dominate those obtained with a model trained on a single-objective optimization.

We explored datasets that are frequently used in recommender systems research, all related to digital media content. Digital media content is consumed frequently and generally without much repetition. The importance of recency and capturing transient behavioral trends may not be equivalent in other recommender systems applications, such as grocery or clothes shopping. The influence of temporal dynamics on these sectors is an exciting topic, and we leave it to future academic and commercial research.

References

- [1] K. Zhou, H. Wang, W. X. Zhao, Y. Zhu, S. Wang, F. Zhang, Z. Wang, J.-R. Wen, S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization, *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (2020)*. URL: <http://dx.doi.org/10.1145/3340531.3411954>. doi:10.1145/3340531.3411954.
- [2] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.
- [3] D. Liang, R. G. Krishnan, M. D. Hoffman, T. Jebara, Variational autoencoders for collaborative filtering, in: *Proceedings of the 2018 world wide web conference*, 2018, pp. 689–698.
- [4] D. Antognini, C. Musat, B. Faltings, Interacting with explanations through critiquing, in: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, (IJCAI 2021)*, 2021.
- [5] P. G. Campos, F. Díez, I. Cantador, Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols, *User Modeling and User-Adapted Interaction* 24 (2014) 67–119.
- [6] N. Milojkovic, D. Antognini, G. Bergamin, B. Faltings, C. Musat, Multi-gradient descent for multi-objective recommender systems, *Proceedings of the AAAI (2020) - Workshop on Interactive and Conversational Recommendation Systems (WICRS) (2020)*.
- [7] B. Marlin, *Collaborative filtering: A machine learning perspective*, University of Toronto Toronto, 2004.
- [8] J. S. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, UAI'98*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998, p. 43–52.
- [9] X. Ning, G. Karypis, Slim: Sparse linear methods for top-n recommender systems, in: *2011 IEEE 11th International Conference on Data Mining, IEEE*, 2011, pp. 497–506.
- [10] Y. Wu, C. DuBois, A. X. Zheng, M. Ester, Collaborative denoising auto-encoders for top-n recommender systems, in: *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, 2016, pp. 153–162.
- [11] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, *arXiv preprint arXiv:1205.2618* (2012).
- [12] C. C. Aggarwal, et al., *Recommender systems*, volume 1, Springer, 2016.
- [13] M. Quadrona, P. Cremonesi, D. Jannach, Sequence-aware recommender systems, *ACM Computing Surveys (CSUR)* 51 (2018) 1–36.
- [14] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, P. Jiang, Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer, in: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1441–1450.
- [15] W.-C. Kang, J. McAuley, Self-attentive sequential recommendation, in: *2018 IEEE International Conference on Data Mining (ICDM)*, IEEE, 2018, pp. 197–206.
- [16] B. Hidasi, A. Karatzoglou, Recurrent neural networks with top-k gains for session-based

- recommendations, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 843–852.
- [17] Y. Ding, X. Li, M. E. Orlowska, Recency-based collaborative filtering, in: Proceedings of the 17th Australasian Database Conference-Volume 49, 2006, pp. 99–107.
- [18] J. Vinagre, A. M. Jorge, J. Gama, Collaborative filtering with recency-based negative feedback, in: Proceedings of the 30th Annual ACM Symposium on Applied Computing, 2015, pp. 963–965.
- [19] A. Chakraborty, S. Ghosh, N. Ganguly, K. P. Gummadi, Optimizing the recency-relevancy trade-off in online news recommendations, in: Proceedings of the 26th International Conference on World Wide Web, 2017, pp. 837–846.
- [20] P. M. Gabriel De Souza, D. Jannach, A. M. Da Cunha, Contextual hybrid session-based news recommendation with recurrent neural networks, *IEEE Access* 7 (2019) 169185–169203.
- [21] X. Luo, Y. Xia, Q. Zhu, Incremental collaborative filtering recommender based on regularized matrix factorization, *Knowledge-Based Systems* 27 (2012) 271–280.
- [22] S. Vargas, P. Castells, Rank and relevance in novelty and diversity metrics for recommender systems, in: Proceedings of the fifth ACM conference on Recommender systems, 2011, pp. 109–116.
- [23] M. Ge, C. Delgado-Battenfeld, D. Jannach, Beyond accuracy: evaluating recommender systems by coverage and serendipity, in: Proceedings of the fourth ACM conference on Recommender systems, 2010, pp. 257–260.
- [24] C.-L. Huang, M.-C. Chen, W.-C. Huang, S.-H. Huang, Incorporating frequency, recency and profit in sequential pattern based recommender systems, *Intelligent Data Analysis* 17 (2013) 899–916.
- [25] D. Kowald, S. C. Pujari, E. Lex, Temporal effects on hashtag reuse in twitter: A cognitive-inspired hashtag recommendation approach, in: Proceedings of the 26th International Conference on World Wide Web, 2017, pp. 1401–1410.
- [26] J.-A. Désidéri, Multiple-gradient descent algorithm (mgda) for multiobjective optimization, *Comptes Rendus Mathématique* 350 (2012) 313–318.
- [27] O. Sener, V. Koltun, Multi-task learning as multi-objective optimization, in: *Advances in Neural Information Processing Systems*, 2018, pp. 527–538.
- [28] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Application of dimensionality reduction in recommender system-a case study, Technical Report, Minnesota Univ Minneapolis Dept of Computer Science, 2000.