Recurrent Intensity Modeling for User Recommendation and Online Matching

Yifei Ma [1] **Ge Liu** [1] **Anoop Deoras** [1]

Abstract

Many applications such as recommender systems (RecSys) are built upon streams of events, each associated with a type in a large-cardinality set and a timestamp in the continuous domain. To date, most applied work is focused on the prediction of the type of the next event, i.e., which exact item a user may visit when they arrive at the RecSys. Instead, we aim to predict when and how often an event of a certain type will be visited by the given user, without the implicit assumption that they will arrive and consume exactly one item at a time. This perspective leads to unique applications in user recommendation (UserRec), where the RecSys is tasked to preemptively match users on behalf of the item producers for marketing purposes. We propose Recurrent Intensity Models (RIMs) that incorporate user visitation intensities in the RecSys, based on recent progress in temporal processes. To our knowledge, our work is the first to approach UserRec completely based on hidden temporal representations without heuristics from explicit feature engineering.

1. Introduction

Many ML applications today involve decision making based on streams of events. In recommender systems (RecSys), reasoning on the stream of events in a user's past history has allowed real-time user-based item recommendations (ItemRec), leading to significant impacts (Hidasi et al., 2016; Hidasi and Karatzoglou, 2018; Ma et al., 2020; Chen et al., 2019; Kang and McAuley, 2018). However, being user-centric, ItemRec has limitations in creating a globally inclusive environment for the item producers. For example, ItemRec tends to over-expose a few popular items to improve the immediate user satisfaction, yet this causes the select items to get even more popular, discouraging the other producers from creating seminal items with higher potential values in the future. This is an important limitation because, as a RecSys involves the participation of both users and item producers, we must consider the success of the item producers for the long-term richness of the RecSys environment. We thus ask this question:

How can we revert a sequence-based RecSys to achieve User Recommendation (UserRec) on behalf of an item producer and, further, balance UserRec and ItemRec by optimizing a surrogate joint objective, which we formulate via Online Matching (OnlnMtch) (Mehta, 2013)?

The UserRec problem aims to evaluate users based on their affinity to a given item or item type, which we call a *type* as a unified name, in a future time window. The affinity scores may be used for offline advertisements. E.g., it helps an item producer choose a segment of users to send marketing emails every morning and then measures their engagements in the next 24 hours. The scores may also be used for online advertisements within a user browsing session that may conclude at an indefinite time. In this case, the item producer must decide in real-time whether they want their items promoted, without any guarantees that the user will continue to engage at all. The item producer has a global constraint that only a fraction of all users can be reached in expectation. Figure 1 shows the UserRec setup in both scenarios, where the online case is mathematically equivalent by changing the window boundaries to relative times while keeping the window sizes comparable.

The UserRec problem is challenging because, unlike ItemRec, each user is:

- *Unique.* Each user consumes at most one item at a time. This is unlike ItemRec, where a popular item can be simultaneously consumed by everyone.
- *Unidentifiable.* There are often magnitude-of-order more users than unique items. In online settings, the users are sampled from a distribution.
- *Evolving over Time.* A past event may stimulate further engagements, whereas sustained inactivity leads to churns. We must redefine user identity to address *Ship of Theseus Paradox: whether an object that has had all of its components replaced remains fundamentally the same object.*

We note that some of these problems are also present in ItemRec in less obvious ways. For example, the state of an item may change after a price change or a phenomenal

[1]Amazon Web Service. Correspondence to: Yifei Ma <yifeim@yma.io>.

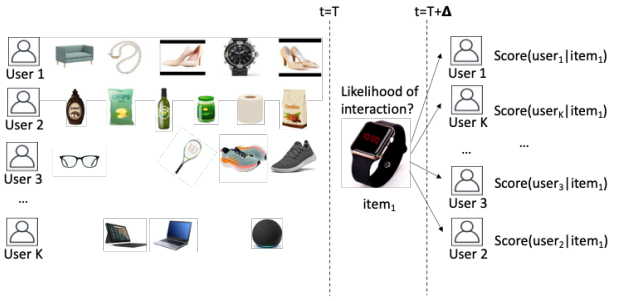


Figure 1: UserRec scenario. Users are considered positive if they indeed interact with the given item in a future time window. A matching hit requires both relevance and activeness of the user. An active user may have multiple hits with different items, whereas an inactive user may have zero hit. In addition, user hits in a given item are often correlated with their past histories with similar items.

review (Wu et al., 2017; Ma et al., 2020). Nonetheless, state changes are often more critical with users than items.

To date, most RecSys have adopted a hidden-state user representation to solve online ItemRec with evolving user states. The user states are based on the user's past event histories, modeled via a family of recurrent neural networks (RNNs), for the prediction of the exact item choice upon their next arrival. However, the ItemRec formulation may not directly translate to UserRec, because the item choice is formulated as a user-conditional distribution without an identifiable *user prior*. To make matters worse, we must compare users with different history lengths for their likelihood of future engagements, which is otherwise omitted in ItemRec because it serves one user at a time.

The cornerstone in UserRec is to connect the user prior with the rate of user engagement events. Note that the user prior does not directly come from probability *density* estimation. Instead, it comes from the conditional *intensity* parameter of a temporal point process based on the recurrent hidden state. Intensity is a characterization of future event predictions by the expected number of events in unit time, whereas density is a summary of the observed events in the past. In this respect, it is also important to distinguish between a general user time and an observed user-event time, which are interchangeable concepts in ItemRec. Like ItemRec, UserRec also considers user states that evolve in time, i.e., treating a different user-time as a different instance. However, unlike ItemRec, a user-time state may not associate with an event. Instead, it sets the initial condition of a future time window, where the hidden state continues to evolve and emit sporadic events as a random process. UserRec looks for the highest expected number of events associated with an item in a comparable time window. As the window size approaches zero, the number of events (in general or associated with the given item type) divided by the window size converges to the intensity parameter, conditioned on the initial state in the user-time instance (Reinhart, 2018).

We propose *Recurrent Intensity Modeling (RIM)* to address the problem of UserRec by intensity estimation of a given type in combination with recurrent state representation. To our knowledge, this is the first work to approach both offline and online UserRec completely based on hidden temporal representations, without explicit feature engineering. It extends two families of work: offline matrix factorization (MF) (Rendle et al., 2009) and online marketing based on customer life-time value (CLV) estimation (Fader et al., 2005). MF decomposes a static matrix of users and items, creating latent vectors that may be used in either UserRec or ItemRec. However, the latent vectors cannot be transferred to unseen users or evolve according to existing users' updates in online scenarios. On the other hand, online CLV utilizes explicit feature designs such as Recency, Frequency, and Monetary value (RFM). While these features may be updated online, CLV cannot leverage the power of latent representation via low-rank decomposition to reduce statistical variance. Our models encompass the benefits in both worlds and are further inspired by recent advances in Neural-TPPs (Mei and Eisner, 2017; Zuo et al., 2020; Shchur et al., 2019) and Neural-ODEs (Chen et al., 2018; Jia and Benson, 2019; Chen et al., 2020). For broader impacts, we novelly connect ItemRec and UserRec in a joint OnlnMtch problem, to further improve satisfaction on both sides.

2. Problem Formulation

We are interested in modeling a temporal process represented by a series of event times and types of a given user. Let $D_u = [(t_j, y_j) : \forall t_1 < t_2 < \cdots < t_j < \ldots, \forall y_j \in A]$ be the random process of this user with a continuous time variable and a type variable in a large-cardinal set A such that $|A| = n$. We consider a user-time as a unique instance $D_u(t) = \{(t_j, y_j) \in D_u : t_j < t\}$ based on its realization until time t. The user-time instance is further summarized by a hidden state representation $h(t) = h(t; D_u)$. We omit D_u when it is obvious from context.

Traditional RNN-based ItemRec forsakes the prediction of the arrival time and only builds recurrent states right before the next event time t_*. It then predict the type choice by a conditional categorical distribution:

$$p(Y = y_*|h(t_*)) = \frac{\tilde{\lambda}_{y_*}(h(t_*))}{\sum_{y \in A} \tilde{\lambda}_y(h(t_*))}, \quad (1)$$

$$\text{where } \tilde{\lambda}_y(h(t_*)) = \exp\left(w_y^\top h(t_*) + b_y + m(h(t_*))\right),$$

where w_y is a row in the decoder weight, which is also interpreted as the hidden embedding of a specific item type, b_y is a global bias in type choices, and $m(h(t_*))$

is a *user-prior* term based on their hidden state. We call $\tilde{\lambda}_y(h(t_*))$ the *unnormalized* probability score and $\tilde{\lambda}_A(h(t_*)) = \sum_{y\in A} \tilde{\lambda}_y(h(t_*))$ the partition function. Note that the value of $m(h(t_*))$ will be canceled out in the categorical distribution. Without identifying this user-prior term, the unnormalized scores cannot be compared across users.

2.1. Recurrent Intensity Modeling (RIM)

With the user-conditional type distribution (1), it is difficult to compare different user-states from different sequences corresponding to unique user-time instances. Instead, we aim to compare sequences by their expected number of events in the future horizon. Denote ΔN_y to be the total number of events of a specific type in the next time window $(t, t + \Delta t]$. For sufficiently small time windows, we may alternatively model the total count of an event type as an independent Poisson distribution, parametrized by a function of the observed events and future window size, $\bar{\Lambda}_y = \bar{\Lambda}_y(t, t + \Delta t; h(t))$:

$$(\Delta N_y | h(t)) \sim \text{Poisson}\left(\bar{\Lambda}_y\right) \quad (2)$$
$$\text{s.t. } \log p(\Delta N_y = n; \bar{\Lambda}_y) = n \log(\bar{\Lambda}_y) - \bar{\Lambda}_y - \log(n!).$$

Proposition 1 (Poisson-Multinomial Connection)**.** *Suppose* $\{N_y \sim \text{Poisson}(\bar{\Lambda}_y) : y \in A\}$ *is a set of independent Poisson random variables. Then, conditioned on the number of total events* $n_A = \sum_{y\in A} N_y$*, the distribution of each variable follows a multinomial distribution* $(N_y | n_A) \sim \text{Mult}(n_A, p)$*, where* $p_y = {}^{\bar{\Lambda}_y}/_{\sum_{\tilde{y}} \bar{\Lambda}_{\tilde{y}}}$*. The categorical distribution in* (1) *is a special case with* $n_A = 1$*.*

Proposition 1 shows that we may alternatively model the RNN-ItemRec predictor (1) as a conditional distribution from independent Poisson priors. Taking the limit $\Delta t \to 0$, the Poisson priors can be more accurately described as temporal point processes (TPPs) (Reinhart, 2018):

$$\lambda_y(h(t)) \sim \lim_{\Delta t \to 0} \mathbb{E}\left[\frac{\Delta N_y}{\Delta t}\middle| h(t)\right] = \mathbb{E}\left[\frac{\mathrm{d}N_y}{\mathrm{d}t}\middle| h(t)\right], \quad (3)$$

which compares to the $\tilde{\lambda}_y$ notation in (1) and to (2) by $\bar{\Lambda}_y = \mathbb{E}\left[\int_t^{t+\Delta t} \lambda_y(h(\tau))\,\mathrm{d}\tau\right]$. The expectation is over the unobserved events in the future horizon, which may require additional calibration models if we directly learn $\lambda_y(h(t))$.

TPP Likelihood. TPPs are more commonly directly learned in $\lambda_y(h(t))$. To do so, we break Δt into infinitesimal windows and join the likelihoods (2) to describe the observation of the eventful or event-less past. Concretely, between any start time t and the next event of type y at $T_y = t_*$, we observe one positive window with count one and all other windows with count zeros, yielding:

$$\log p(T_y = t_* | D_u) = \log \lambda_y(h(t_*; D_u)) - \Lambda_y(t, t_*; D_u),$$
$$\text{where } \Lambda_y(t, t_*; D_u) = \int_t^{t_*} \lambda_y(h(\tau; D_u))\,\mathrm{d}\tau. \quad (4)$$

Notice the subtle difference that the Λ_y in (4) is one realization in the expectation for $\bar{\Lambda}_y$, because the likelihood is based on fully observable data until $D_u(t_*)$. (Thus the need to calibrate for $\bar{\Lambda}_y$ after learning the instantaneous λ_y.) The numerical integration may be solved by importance sampling (Mei and Eisner, 2017), ODE solvers (Chen et al., 2018), or analytical forms in the case of Hawkes processes.

RIM for Sets. Thanks to Proposition 1, we may extend (2) and (4) to subsets of types as $\bar{\Lambda}_S$ and λ_S, $\forall S \subseteq A$.

2.2. Applications

RIM allows UserRec based on their affinity to a given item type y. The decision is in the space of user-time that is appropriate for the application, e.g., all users at a given time in offline marketing. We call this space $H = H(U, T)$ to be consistent with user-time state representation. UserRec then solves for a vector of assignment $\pi(\cdot; y) : H \to \{0, 1\}$ under a finite budget $C(y) \le |H|$:

$$\max_{\pi(h;y)\in\{0,1\}} \sum\nolimits_{h\in H} [\pi(h; y)\lambda_y(h)] \quad (5)$$
$$\text{s.t. } \sum\nolimits_{h\in H} \pi(h; y) \le C(y).$$

For online marketing, the user states are drawn from a distribution $P(H)$. We may replace the summation in (5) with Lebesgue integral in the probability space. In fact, we further extend the assignment problem to consider constraints in both UserRec and ItemRec for an OnlnMtch solution $\pi : H \times A \to \{0, 1\}$ that maximizes the global objective

$$\max_{\pi(h,y)\in\{0,1\}} \iint [\pi(h, y)\lambda_y(h)]\; \mathrm{d}P_H(h)\,\mathrm{d}P_A(y) \quad (6)$$
$$\text{s.t. } \begin{cases} \int \pi(h, y)\,\mathrm{d}P_A(y) \le k(h), & \forall h \in H; \\ \int \pi(h, y)\,\mathrm{d}P_H(h) \le c(y), & \forall y \in A; \end{cases}$$

where $k(h)$ and $c(y)$ are user and item type capacities in relative terms. Choosing $k(h) = 1$ and $c(y) = {}^{C(y)}/_{|H|}$ reduces to the offline UserRec problem (5). Choosing $k(h) = {}^{K}/_{|A|}$ and $c(y) = 1$ reduces to the traditional Top-K ItemRec problem. In its most general formulation, OnlnMtch (6) simultaneously solves for UserRec and ItemRec, achieving global diversity by avoiding over-exposures of already popular item types and over-concentration to busy users.

3. Proposed Models

While RIM is motivated to estimate the intensity of each type separately, we find it more convenient to utilize ex-

isting RNN-ItemRec models. Based on Proposition 1, we keep the RNN model for its categorical prediction of the preference *direction*, $p(y|h(t))$, while introducing separate estimators for the global user intensity *norm*, $\mathbb{E}[\Delta N_A|h(t)]$, and estimate the corresponding λ_A and $\bar{\Lambda}_A$ by RIM for sets.

We emphasize that RNN-TPPs are able to recover the intensity scores in each item type, due to the law of total expectation as a corollary of Proposition 1: $\mathbb{E}[\Delta N_y|h(t)] = \mathbb{E}[\mathbb{E}(\Delta N_y|\Delta N_A)|h(t)] = p(y|h(t))\mathbb{E}[\Delta N_A|h(t)]$, assuming relatively stable preference-direction predictions that are independent of total event counts.

RNN-Pop is our simplest model, where we use the length of user histories from the collected data. This approach works due to a homogeneous intensity assumption, i.e., the number of past events directly correlates with the future event intensity. It also assumes that all user activities are collected over a comparable period of time, which coincidentally holds because practical RecSys often truncates the training datasets by a fixed time for scalability concerns.

RNN-Hawkes. To extend the naive RNN-Pop models, we may break the homogeneity assumption to consider user state changes that affect future event intensity. In this regard, Hawkes process assumes a positive stimulation through past user events and a gradual churn-out effect after sustained inactivity. It models these effects via an exponentially-decaying kernel in the intensity parameters

$$\lambda_A(t) = \mu + \sum_{j:t_j<t} \sum_{r=1}^{R} \alpha_r \phi_r(t - t_j), \tag{7}$$

$$\text{where } \phi_r(t) = \tfrac{1}{s_r} \exp\left(-\tfrac{t}{s_r}\right) 1_{t>0}, \text{ and } \mu, \alpha, s > 0.$$

Here, we extend it with a mixture of R latent kernels with learned coefficients and learn it with *tick* software package (Bacry et al., 2017). We design the R kernels to have log-spaced half-lives between 10^{-3} and $10t_{\max}$ where $t_{\max}$ is the largest temporal span in user histories. A large-s_r kernel approximates the RNN-Pop model with a step function after each observation, whereas a small-s_r kernel suggests a fast-diminishing effect, often within a short browsing session.

RNN-Hawkes-Poisson. For long prediction horizons, we may also calibrate Hawkes scores to eliminate the effects of fast-diminishing kernels. Notice that the Hawkes states $\tilde{h}(t)$ are already positive. We thus model their weighted sums as

$$\bar{\Lambda}_A = \tilde{h}(t) \odot \text{softplus}(w) = \tilde{h}(t) \odot \log(1 + \exp(w)). \tag{8}$$

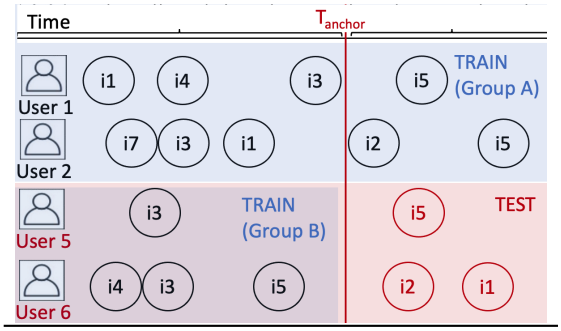
We further concatenate $\tilde{h}(t)$ with the RNN partition function $\tilde{\lambda}_A(h(t))$, which we observe to be positively correlated with the future user intensity. Our additive formulation is inspired by (Mei and Eisner, 2017) and we slightly improve it by allocating the softplus to each coordinate to improve numerical stability.

4. Experiments

We conduct experiments in three RecSys datasets with unique properties. We set $k(h) = c(y) = 1\%$ in the respective UserRec and ItemRec problems.

- **Netflix (NF)**[1] is a movie rental service where movie ratings from a user's past affects their personalized ItemRec in the future. NF is widely used in rating prediction and ItemRec, but less so in UserRec and OnlnMtch.
- **Movielens (ML)** (Harper and Konstan, 2015)[2] is a movie rating website that collects user preferences to study RecSys. Over time, it released several versions of datasets. We use a small-scale ML-1M in an 2003 release with one million rating events between six thousand users and three thousand items to easily validate our ideas.
- **Yoochoose (YC)**[3][4] is a dataset used in RecSys 2015 Challenge with six months of activities for product recommendation such as general tools, toys cloths, electronics, and much more.

4.1. Data Splits



	NF	ML	YC
# warm users	32238	3020	71784
# warm items	16217	3669	11431
# training events	2437151	762016	1087267
# clean test events	187096	37597	49500
UserRec C(y)	323	31	718
ItemRec K(h)	163	37	115

Table 1: Data split and statistics

We aim to split the data in a way to conduct evaluation within a time window. On NF, we simply pick a test window between 2005/6/15 and 2005/6/28 with a training window between 2005/1/1 and 2005/6/14. On ML and YC, however, users typically have short browsing sessions that do not overlap. Instead, since we treat every user-time as an instance,

[1] https://www.kaggle.com/netflix-inc/netflix-prize-data
[2] https://grouplens.org/datasets/movielens/1m/
[3] https://www.kaggle.com/phhasian0710/yoochoose
[4] https://recsys.acm.org/recsys15/challenge/

Table 2: RecSys methods evaluated by ranking relevance (precision) and diversity (target perplexity). RNN-TPP models are clear winners in both ItemRec and UserRec. They also generate larger diversity compared with non-personalized baselines.

ItemRec (Top-K Items per User)							UserRec (Top-C Users per Item)						
Model	Precision ($\times 100$)			Item Perplexity			$\times$ User Intensity	Precision ($\times 100$)			User Perplexity		
	NF	ML	YC	NF	ML	YC		NF	ML	YC	NF	ML	YC
Rand	0.04	0.37	0.006	16192	3610	11423	Rand	0.04	0.37	0.006	32140	2980	71472
Pop	0.78	1.74	0.067	163	37	115	Pop	0.19	3.03	0.028			
							Hawkes	0.26	4.44	0.030	323	31	718
							Hawkes-Poisson	0.35	4.48	0.034			
BPR-Item	0.71	0.73	0.222	1139	684	1774	BPR-User	0.26	0.84	0.147	1843	777	18629
RNN	0.86	2.09	0.306	1015	1049	5024	(Uniform)	0.19	1.43	0.223	8910	1952	45992
							Pop	0.28	2.81	0.233	3520	1041	44620
							Hawkes	0.38	5.42	0.235	2268	375	44880
							Hawkes-Poisson	0.38	5.15	0.258	4720	442	34533

we may assign different prediction windows with different starting times to different users, as long as they have comparable horizons. Further, we split the training and testing set by users. Table 1 shows that we first create Group A and Group B with 50% users each, followed by TEST-window creation in Group B according to specific user-start time. Specifically, the relative test-start time comes from the 50% global quantile in the temporal span of all user histories and the test-window size is comparable with the time duration before the test starts.

The RNN-Hawkes-Poisson model requires an additional validation window to calibrate by the aggregate statistics in long horizons. For NF, this happens between 2005/6/1 and 2005/6/14. For MF and YC, we pick the corresponding validation windows among the training users. The statistics of all three datasets may be found on Table 1.

A caveat in data preparation is to avoid temporal leakage. For example, a simple retention of all observed users would be biased, because users with zero history are destined to have nonzero future events, which disagrees with our common prior that inactive users in the past are unlikely to engage in the future. The issue is that we under-represent users with zero history and zero future events. As a remedy, we include only warm users with at least one past event, so that their future engagements are no longer biased. We also retain only warm items for similar considerations. For YC, we set a higher threshold with at least 4 past events per user and 10 past events per item.

4.2. UserRec and ItemRec Results

Table 2 shows results from the baselines and our proposed methods, where RNN-TPP models are clear winners in ranking metrics such as Precision@Top-1% in both UserRec and ItemRec scenarios. Further, in ItemRec, we observe a trend in Rand < Pop < BPR < RNN, meeting our expectations. On UserRec, we also see contributions from both the prediction of *preference directions* and user *engagement priors*. For preference directions, we expect Rand < Pop < BPR < RNN-Pop. For engagement priors, we expect a significant contribution in RNN-Pop models over the RNN (non-prior) baseline. Additionally, we hope to observe Pop < Hawkes < Hawkes-Poisson as we build more complex TPP models. We see the TPP model improvements obvious in NF and ML but less so in YC. This is because complex TPPs help most when users have denser histories.

Besides delivering higher relevance, personalized models also have the side benefit to increase the global diversity in the recommended targets, which can be measured with

$$\text{perplexity} = \exp\left(-\sum\nolimits_{y\in A} \tilde{p}(y)\log(\tilde{p}(y))\right),$$
$$\text{where } \tilde{p}(y) = \frac{\text{total \# of Item-}y\text{ recommended}}{\text{total \# of all recommendations}},$$

or its counterpart in UserRec. Higher diversity is often desirable because it improves the exposure of less-popular items across different user requests or ensures a balanced distribution among the recommended users when coordinating different item producers. A perplexity score of X shows that the global uncertainty in the target distribution, marginalized over all recommendation requests, is comparable to a fair X-sided dice. Popularity-based models are fixed and contain the least amount of uncertainty. On the other hand, learning-based models yield significantly larger perplexity. The increased perplexity allows these models to reach more diversified user pools to show larger potential impacts than what is reflected in the relevance score comparisons.

4.3. Online Matching (OnlnMtch) Results

Previously, people study ItemRec and UserRec separately, without realizing that one affects the other. To further explore the diversity effects of personalized models, we for-

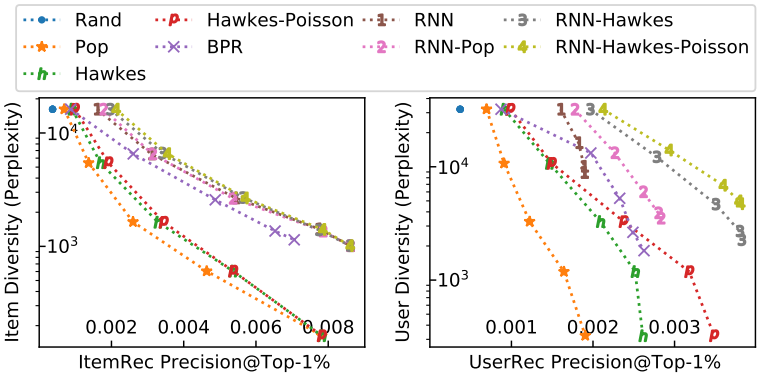


Figure 2: OnlnMtch solves for recommendations under global constraints in the targets. Treating the original ItemRec and UserRec as unconstrained optimization, we bound the total numbers of times each item and user can be be recommended, respectively, to increase the global diversity in the targets. We show the Pareto fronts of diversity-relevance trade-offs for all proposed models, where RNN-TPPs dominate. Showing results on Netflix dataset.

mulate the problem in (6) that combines Top-K constraints in ItemRec and Top-C constraints in UserRec. On Figure 2(left), we keep $k(h) = 1\%$ of the total number of items, i.e., to recommend Top-163 personalized items to every user, while adjusting the global item exposure capacities by $c(y) \in [1\%, 3\%, 10\%, 30\%, 100\%]$. The $c(y) = 1\%$ limit is set to create an exact one-to-one matching condition, i.e., $k(h) = c(y)$, and the $c(y) = 100\%$ condition agrees with ItemRec with unbounded item exposures. As we lower $c(y)$, we tune-down the exposure rate of popular items.

The OnlnMtch problem is solved by greedy assignment, where we first sort all the estimated expectations $\lambda_y(h)$, followed by greedy assignments of the corresponding user-item pairs while the capacities remain unsaturated. We also experimented with linear programming for global optimality, but we did not find significant improvements and thus adopt the greedy solution for its simplicity. On Figure 2(right), we repeat the experiment for UserRec, with fixed $c(y) = 1\%$ and varying $k(h) \in [1\%, 3\%, 10\%, 30\%, 100\%]$, for increased freedom in user selection.

From the results, we see that all methods have impacted relevance under global constraints in the targets. Comparing the Pareto fronts, we see clear winners from RNN-Hawkes and RNN-Hawkes-Poisson. More generally, personalized models (BPR and RNN) perform better than unpersonalized models and RNN models further outperform BPR models. Among each subgroup, TPP models (Hawkes and Poisson) outperform vanilla Popularity counts. The difference becomes larger as the balance of the constraints focus more on the item capacity limitations.

In a sense, Figure 2(right) is a continuation of Figure 2(left) where the ratio of $c(y)/k(h)$ keeps decreasing. The left-top points of the two plots agree in user-item matching assignments, because the user and item constraints are both tight. On the other hand, item diversity hurts ItemRec relevance more than user diversity hurts UserRec relevance, perhaps showing that UserRec has larger design freedom. UserRec may be the new frontier of research.

5. Conclusion

We study UserRec in an online RecSys paradigm, where each user is represented by their history of item consumption events, without revealing their unique identities. This problem is of great values in keeping the diversity and liveliness of RecSys; by addressing the pain points in item marketing, the item creators can focus on their expertise in creation.

Our main challenge is to discover the missing user-marginal likelihood to receive new product promotions, in addition to the prediction of their preference directions. We novelly connect this challenge to the intensity parameter estimation of a temporal point process. We explain the key difference between probability density estimation of an observed user event and temporal intensity prediction within a user time-window. In our proposal, the user state is interpreted as a stochastic process that continuously evolves with or without explicit update by observable events. We leverage recent progress in TPPs to build concise models. Our models evaluate well in both UserRec and ItemRec on three RecSys datasets that are previously used only for ItemRec.

RecSys is an area of broad social impacts. The techniques we develop are aimed to improve the diversity and welfare of all participating parties. We use global optimiality as a surrogate in OnlnMtch reformulation of RecSys. We show promising connections between RIM and better diversity-relevence trade-offs.

References

E. Bacry, M. Bompaire, S. Gaïffas, and S. Poulsen. tick: a Python library for statistical learning, with a particular emphasis on time-dependent modeling. *ArXiv e-prints*, July 2017.

Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H. Chi. Top-k off-policy correction for a reinforce recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018.

Ricky TQ Chen, Brandon Amos, and Maximilian Nickel. Neural spatio-temporal point processes. In *International Conference on Learning Representations*, 2020.

Peter S. Fader, Bruce G. S. Hardie, and Ka Lok Lee. Rfm and clv: Using iso-value curves for customer base analysis. *Journal of Marketing Research*, 42:415 – 430, 2005.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

Balázs Hidasi and Alexandros 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*, pages 843–852, 2018.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks, 2016.

Junteng Jia and Austin R Benson. Neural jump stochastic differential equations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, 2018.

Yifei Ma, Balakrishnan (Murali) Narayanaswamy, Haibin Lin, and Hao Ding. Temporal-contextual recommendation in real-time. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20, 2020.

Aranyak Mehta. Online matching and ad allocation. *Foundations and Trends® in Theoretical Computer Science*, 8 (4):265–368, 2013.

Hongyuan Mei and Jason Eisner. The neural hawkes process: a neurally self-modulating multivariate point process. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

Alex Reinhart. A Review of Self-Exciting Spatio-Temporal Point Processes and Their Applications. *Statistical Science*, 33(3):299 – 318, 2018. doi: 10.1214/17-STS629.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, page 452–461, 2009.

Oleksandr Shchur, Marin Biloš, and Stephan Günnemann. Intensity-free learning of temporal point processes. In *International Conference on Learning Representations*, 2019.

Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. Recurrent recommender networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, 2017.

Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer hawkes process. In *International Conference on Machine Learning*, pages 11692–11702. PMLR, 2020.