Accordion: A Trainable Simulator for Long-Term Interactive Systems

James McInerney
jmcinerney@netflix.com
Netflix
Los Gatos, CA, USA

Ehtsham Elahi
eelahi@netflix.com
Netflix
Los Gatos, CA, USA

Justin Basilico
jbasilico@netflix.com
Netflix
Los Gatos, CA, USA

Yves Raimond
yraimond@netflix.com
Netflix
Los Gatos, CA, USA

Tony Jebara
tonyj@spotify.com
Spotify & Columbia University
New York, NY, USA

ABSTRACT

As machine learning methods are increasingly used in interactive systems it becomes common for user experiences to be the result of an ecosystem of machine learning models in aggregate. Simulation offers a way to deal with the resulting complexity by approximating the real system in a tractable and interpretable manner. Existing methods do not fully incorporate the interactions between user history, recommendation quality, and subsequent visits. We develop Accordion, a trainable simulator based on Poisson processes that can model visit patterns to an interactive system over time from large-scale data. New methods for training and simulation are developed and tested on two datasets of real world interactive systems. Accordion shows greater sensitivity to hyperparameter tuning and offline A/B testing than comparison methods, an important step in building realistic task-oriented simulators for recommendation.

CCS CONCEPTS

• **Computing methodologies** → **Simulation environments**; • **Information systems** → **Recommender systems**.

KEYWORDS

Poisson Process, Deep Learning, Simulation

ACM Reference Format:
James McInerney, Ehtsham Elahi, Justin Basilico, Yves Raimond, and Tony Jebara. 2021. Accordion: A Trainable Simulator for Long-Term Interactive Systems. In *Fifteenth ACM Conference on Recommender Systems (RecSys '21), September 27-October 1, 2021, Amsterdam, Netherlands.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3460231.3474259

1 INTRODUCTION

Consider a machine learning practitioner who is building or improving a recommender system. After obtaining a dataset of features and responses from historical interactions, she trains a model of user preference to improve personalization within an interactive system of users, items, and models over time. She may be replacing one or several of the models at various positions or components of the system while leaving the remaining models unchanged.[1]

Before deploying the new model, the practitioner must confront a series of questions. What will be the overall effect of replacing the model in component k with a new model? Such effects include the incremental benefit of the behavior of the model at k with respect to the behavior of all other models $j \neq k$ as well as the change in dataset collection that results from the altered system. What downstream interactions will the new model create, either directly with the user or indirectly via other models? If there are hyperparameters influencing data collection (e.g., exploration rates, post-hoc business logic) how are they to be optimized? More fundamentally, what is the mechanism of the anticipated improvement and is it within acceptable bounds? For example, it is widely known that showing familiar items based on recent interactions increases short-term click rate [16], but a recommender that shows only recent items will not help a user discover new items, which is important in the long-term. If the improvement of a proposed system operates via mechanisms that create negative externalities, e.g. increasing short-term metrics at cost of the long-term outcomes, that are not accounted for by measured quantities (either offline or in an A/B test) then the proposed change may do more harm than good in the totality of the interactive system.

The challenge inherent to answering these questions is a result of the fact that, in application, machine learning systems tend to coexist in a shared ecosystem. For instance, a user experience is often the result of numerous machine learning systems layered iteratively atop each other over a multi-year time period. These models may span a variety of paradigms from unsupervised, supervised, semisupervised and reinforcement learning as well as methodologies from deep learning, Bayesian modeling, and more. Due to such heterogeneity, it can eventually become impossible to reconcile them into a single conceptual model. Even simple systems interact with users and other stakeholders in complex ways. As the system approaches a certain level of complexity, it becomes more feasible to simulate it.

Recent research in simulation has presented platforms for building new simulators based on abstract assumptions about users and

[1]For example, changing a user-item affinity scoring model whose output is used in some, but not all, downstream models affecting item impressions.

RecSys '21, September 27-October 1, 2021, Amsterdam, Netherlands

ACM ISBN 978-1-4503-8458-2/21/09…$15.00
https://doi.org/10.1145/3460231.3474259

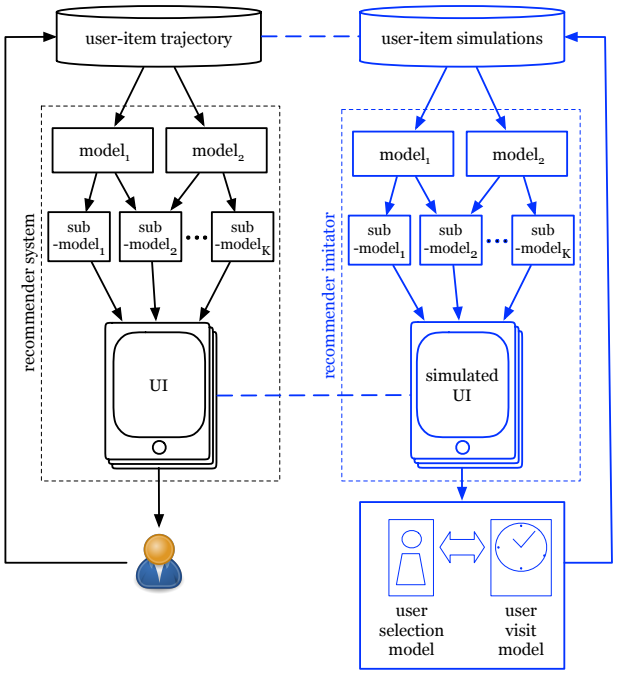


Figure 1: Diagram showing how the simulator is organized. The generative process for observational trajectories appears on the left in black and the simulator appears on the right in blue. There are two points of interaction between observational and simulated data indicated by the dashed lines, one between databases of user-item interactions and the other between logged UI impressions.

items [11, 26]. While it is possible to recreate emergent pathologies from such simulators, it does not answer the question of how any specific real-world system behaves, nor does it answer questions related to incremental improvement and optimal settings. In a limited sense, these quantities can be approximated with offline evaluation on historical datasets using the new model in isolation. But, with the proliferation of models in modern infrastructure, no single model acts in isolation. In this way, the practitioner's goal of improving individual models encounters a coordination problem with other components of the system.

Furthermore, a key aspect of modeling the entire system is the pattern of user visits. Extant methods do not include a generative account of visit behavior. The choice of visit time patterns reveals information about the user state, e.g., satisfaction or curiosity, that are dependent on the quality of previous recommendations. A full account of the emergent properties of an interactive system includes the effect that previous interactions (and their success) have on visits.

To address these shortcomings we develop Accordion, a set of novel methodologies for the simulation of interactive systems that minimize the gap between simulation and reality (the *sim2real* gap) and capture the behavior of recommenders and users over time. A high-level design is given in Figure 1. The full interactive system being imitated is highly complex and is shown on the left hand side of Figure 1. It comprises various models, algorithms, user interfaces (UI), user experiences (UX), and associated features that may require extensive computational resources. While it is possible, in principle, to query the non-user components of the real system many times to perform holistic evaluations of the kind previously discussed, the computational cost is prohibitive. The right hand side of Figure 1 is a "good enough" approximation (the simulation) with two points of contact with reality. The first is the logged trajectories of users and items that result from interactions over time. The second is the impressions comprising the output of the recommender system. Taken together, these induce a training objective to optimize with respect to the parameters of the simulation.

We base the methodology of Accordion on the inhomogeneous Poisson process [12], a stochastic process with strong simplifying assumptions that are nonetheless flexible enough to capture emergent properties of interactive systems over time. The key simplifying assumption is that non-overlapping time intervals exhibit statistical independence as it relates to the random variables of the simulation conditioned on its parameters. While this assumption appears overly restrictive at first, particularly for characterizing systems over time, inhomogeneity allows us to capture a large range of behaviors. In more detail, the problem of simulation is formalized into that of learning an *intensity function* which maps arbitrary user and item features to the differential number of events (i.e. visits, interactions) evaluated at specific points in time. The intensity function is tractable to learn from a potentially large number of observed events. Furthermore, the superposition property of the Poisson process enables interpretability of the trained simulator and its data samples. For example, Figure 2 shows the trained simulator for a random user in the ContentWise dataset [21] and visualizes which assumptions contributed to the fit.

The contributions of this work are as follows:

- We present Accordion, a fully trainable simulator for interactive systems based on inhomogeneous Poisson processes that enables the comparison of different realistic simulation settings and their effect on the total number of visits, positive interactions, impressions, and any other empirical quantity that can be derived from a sampled dataset.
- We develop a novel scalable algorithm for training deep inhomogeneous Poisson process models and provide a model architecture that decomposes the simulation problem into a set of simpler components allowing interpretability of sampled events.
- We provide empirical comparisons between Accordion and other approaches to simulating interactive systems and find that the ability to simulate variable-sized datasets is crucial to the end goal for the common tasks of hyperparameter and model selection. We use data from online A/B testing, the gold standard for evaluating algorithms, to verify our approach.

The rest of the paper is organized as follows. We discuss related work in Section 2. Then, we present the simulator in Section 3 and develop model assumptions, the training algorithm, and simulation algorithm. An empirical evaluation in Section 4 with two large real-world datasets is provided comparing the simulator against benchmarks. We discuss conclusions and future work in Section 5

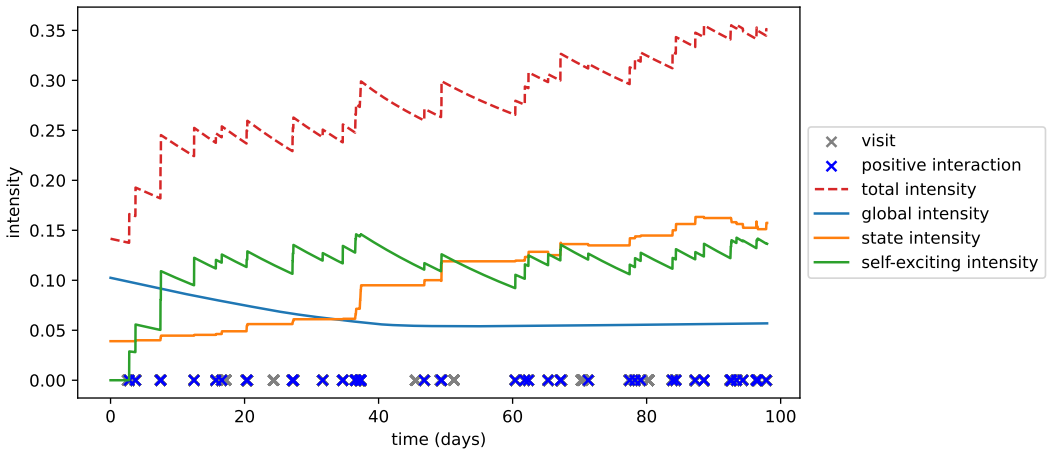


Figure 2: The superposition property of the Poisson process enables interpretability of the trained simulator and its data samples. The plot shows the intensity fit for a random user in the ContentWise dataset [21]. The real observed user visits are plotted with light crosses (visits) and dark crosses (positive interactions). The modeled intensity function is plotted with a dashed line. Total intensity is a sum of the intensities of three sub-components: global, state, and self-exciting intensity which depends on successful recommendations in the history.

2 RELATED WORK

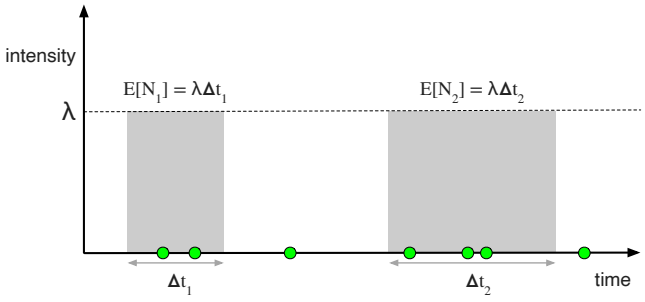
We discuss related work from several areas of research in recommender systems, likelihood-free inference, and deep Poisson process models.

Recommender Systems Simulators. There is a long history of using simulation to evaluate systems for information retrieval, filtering, and recommendation [5, 8, 19]. A comprehensive account is beyond the scope of this paper. Although the types of item interaction have changed significantly, and along with them, fundamental changes in methods for recommendation and evaluation, it has long been recognized that user interactions exist within a larger system with complex outcomes and various stakeholders, e.g., users, producers, information platforms [1]. Simulations are used to evaluate how algorithms improve the user experience and success of the overall system (e.g., purchases on an e-commerce platform, user satisfaction with content) as well as to identify and study, e.g., filter bubbles [6], statistical bias [10]. Other research introduces simulation platforms to encourage collaboration around a wider scope of simulation models and to provide simulated interventional data for reinforcement learning algorithms. RecSim [11] and RecoGym [26] provide frameworks allowing users and items to interact in a discrete-time Markovian fashion, both wrapped in the OpenAI gym interface for reinforcement learning [4].

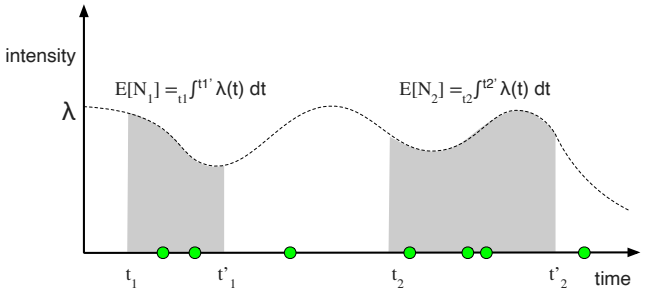
Most simulators are synthetic, in the sense that, although they can provide existence proofs of emergent properties in interactive systems, they are not trained to imitate any specific system from data. An exception is RecSim NG, an extension to RecSim, that provides a platform using probabilistic programming to allow a high degree of flexibility in specifying and training agents and entities in a simulated interactive system, e.g., users, items, content providers [18]. The platform exposes data likelihood under imperatively-specified models allowing inference over the random variables in simulation. The current engine for specifying the likelihood is based on discrete-time Markovian dynamics. While it is possible, in principle, to approximate continuous-time, non-Markovian assumptions and to make missingness explicit, doing so increases the number of "nuisance variables" in the system that fuels variance and makes inference non-trivial.

Likelihood-Free Inference. A wide range of inference methods may be used to train a simulator. They fall into two broad categories, likelihood-based inference and likelihood-free inference. Likelihood-based inference uses a tractable explicit likelihood function to infer the random variables of the simulation. Popular methods include maximum likelihood estimation, maximum *a posteriori*, Markov Chain Monte Carlo, variational inference [2]. The requirement of having an explicit likelihood function restricts the applicable class of models. Likelihood-free inference can apply to the full range of Turing-complete generative processes and uses the discrepancy between simulated data and real data to infer the parameters [15, 27]. It is most useful for existing imperative implementations of complex generative processes, often created by experts in other domains, e.g., evolutionary biology, econometrics, physics [22, 24]. However, there is a lack of theory guiding the design of the discrepancy function, resulting in data-specific heuristic design. Furthermore, the lack of assumed structure means that inference, in general, is more computationally intensive. For these reasons we opt for an explicit likelihood approach based on the Poisson process [12].

Deep Poisson Processes. Like many classical methods from statistics, Poisson processes have undergone adaptation to settings accessible to large scale data and computation in recent years [9]. Extensions to learning the Poisson process intensity function at scale include measure transport [20], variational autoencoders [3],



(a) Homogeneous Poisson process



(b) Inhomogeneous Poisson process

Figure 3: Examples intensity functions for Poisson processes. Area under curve for sub-intervals expresses expected number of events.

stochastic gradient Riemannian Langevin dynamics [13]. In comparison, our approach sidesteps some of the complexity of inference by specifying a fully amortized model, i.e., the generative process samples interaction observations from a parameterized conditional distribution without any latent variables. Furthermore, we focus on formulating a decomposable interpretable intensity function through superposition then fitting it with stochastic gradients evaluated at event times, an approach that may be combined with the aforementioned methods to incorporate more sophisticated uncertainty quantification.

3 INHOMOGENEOUS POISSON PROCESS SIMULATOR

In this section we present Accordion, a methodology for simulating user-item trajectories with a variable number of events. In Section 3.1, we introduce the simulator and its model components. Then, in Section 3.2, we present an efficient training algorithm for the simulator that scales to large recommendation datasets. Finally, in Section 3.3, a scalable algorithm is developed for simulating new trajectories after training.

3.1 Model

A simulated trajectory is a random number of N events consisting of continuous times $t_{1:N} \in [0, T]$, user identifiers $u_{1:N}$, items $i_{1:N}$, and rewards $r_{1:N}$. Rewards may take any form, e.g. scalar or vector, continuous or discrete. Call the simulated dataset $\mathcal{S}$ and the real observed dataset $\mathcal{D}$, both assumed to come from the same generative model. A trainable simulator is one that can fit the set of parameters θ determining the sampled trajectories $\mathcal{S}$ given the data $\mathcal{D}$ and model.

The challenge arising from having the number of observations N be random when training a simulator is that training objectives in most modern machine learning methods take the form $\mathcal{L}_\theta = \sum_{n=1}^{N} l(x_n, \theta)$ where N is a given quantity that is fixed and l is the per-data point contribution to the objective. Yet, as argued in Section 1, the number of observations provides important information. Point processes provide a framework for modeling variable-sized datasets. In a sense, the length of the dataset can expand or contract depending on the distributions used in simulation.

The simplest point process is the Poisson process, which assumes that non-overlapping time intervals have an independent number of events and the number of events is Poisson-distributed [12]. We focus our attention on the 1-dimensional Poisson process to capture changes over time. Figure 3 gives two examples of a Poisson process. For both, the number of events in the range $[0, T]$ for any subinterval (t_1, t_2) is Poisson-distributed $N(t_1, t_2) \sim \text{Poisson}(\int_{t_1}^{t_2} \lambda(t)\text{d}t)$ where λ is the *intensity function* that characterizes numbers of events over time such that $0 \leq \int \lambda(t)\text{d}t \leq \infty$ for all time intervals. Since the mean and variance of the Poisson distribution are equal to its rate, the expected number of events in an interval is also equal to the area under the curve $\mathbb{E}[N(t_1, t_2)] = \int_{t_1}^{t_2} \lambda(t)\text{d}t$. When λ is a constant function, the area becomes stationary, i.e., it depends only on the size of the time interval, a special case known as the homogeneous Poisson process (Figure 3a). The more general case is known as the inhomogeneous Poisson process (IPP) in Figure 3b where the intensity function can be any arbitrary continuous function respecting the non-negative finite area property.

Our goal is then to characterize visit patterns by learning the intensity function λ. Since each user is likely to have a different intensity, we condition the intensity function $\lambda(t \mid \xi)$ on features $\xi \in \mathbb{R}^K$ that capture all relevant user and item information. As we will discuss in Section 3.2, in contrast to standard learning objectives, the Poisson process likelihood depends on features ξ that were not explicitly logged in the dataset $\mathcal{D}$. Therefore, it is more convenient to refer to a featurizing function $x : \mathbb{R}^+ \to \mathbb{R}^K$ which yields features $\xi = x(t)$ for any query time t.[2]

In developing our approach, we use two important properties of the Poisson process: superposition and marking. Overlapping any two independent Poisson processes with intensities λ_1 and λ_2 results in a third Poisson process with intensity $\lambda = \lambda_1 + \lambda_2$, referred to as a superposition of Poisson processes. For simulation, this enables modularity to capture different kinds of structure in the data, e.g., visits driven by user state or recent activity, without exploding the number of parameters in the simulator. It also enables interpretability of model components because the overall intensity is a sum of non-negative outputs of the intensity sub-models. As an example of the kind of modularity and interpretability possible using the superposition property, Figure 2 shows the events from a real user in the ContentWise dataset (more details in Section 4) and the overall model output with its constituent intensity functions.

[2]Since time is continuous in this setting, each unique timestamp maps to a specific event or potential event that may be featurized.

The other key property of the Poisson process is the ability to mark or annotate each event with additional random variables. Specifically, we mark each event by i, the item the user interacted with, r, the outcome of the interaction.

The visit model generates visits with intensity,

$$\lambda(t \mid x(t)) = \lambda_{\text{global}}(t) + \lambda_{\text{state}}(t \mid x(t)) + \lambda_{\text{hawkes}}(t \mid x(t)), \quad (1)$$

where the architectures for each intensity subcomponent are shown in Figure 4 and each intensity is conditional on the set of parameters θ, omitted here to reduce notational burden. The global intensity (Figure 4a) acts across all users and can either be a constant scalar or a function of time. State intensity in Figure 4b takes a user representation (e.g., bag of words of items with positive interactions, sequential summary of previous interactions) and maps it to the intensity associated with that state. Hawkes intensity [23] in Figure 4c boosts the overall intensity after positive interactions up to the query time t,

$$\lambda_{\text{hawkes}}(t \mid x(t)) = \sum_{j<n(t)} a_{i_j} \exp\{-b_{i_j}(t - t_j)\}. \quad (2)$$

for non-negative parameters (a, b) that are indexed by the item that was interacted with at event j. The Hawkes intensity allows for good recommendations to boost the number of future visits.

There is broad scope for further exploring other types of structure in the intensity function, e.g., habitual time structure based on time of day, week, month etc., or exogenous events such as weather or public holidays.

We also require marking distributions $p(i, r \mid x(t))$ that determine which items are presented and interacted with on each visit, i.e., the user selection model. We factorize the joint distribution over item and reward $p(i, r \mid x(t)) = p(i \mid x(t))p(r \mid i, x(t))$ and model each with a softmax, in line with policy imitation in reinforcement learning and autoregressive models of click behavior [14]. Both the impression model and user selection model use as input a bag of words representation of previous user interactions with items and map these though a dense network to a multinomial distribution over items. A summary diagram of the observed random variables in the model are shown in Figure 5. We next consider how to train the model on large-scale data using stochastic gradients.

3.2 Training

The model presented in Section 3.1 has a set of free parameters θ that comprise the weights and intercepts of all the functions that define the overall intensity function λ_θ. The likelihood of the Poisson process induced by λ_θ and the dataset $\mathcal{D}$ is,

$$p(\mathcal{D} \mid \theta) = \frac{\prod_{i=1}^{N} \lambda_\theta(t_i \mid x(t_i))}{\exp\left\{\int_0^T \lambda_\theta(t \mid x(t))\mathrm{d}t\right\}}. \quad (3)$$

Taking a maximum likelihood approach to train parameters θ results in the objective,

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \log \lambda_\theta(t_i \mid x(t_i)) - \int_0^T \lambda_\theta(t \mid x(t))\mathrm{d}t, \quad (4)$$

where the $\theta^* = \arg_\theta \max \mathcal{L}(\theta) = \arg_\theta \max p(\mathcal{D} \mid \theta)$ due to the increasing property of log.

Although the objective in Equation 4 arose naturally from the definition of the Poisson process, it has another intuition in the context of maximum likelihood estimation. The first term in Equation 4 rewards intensity models that place high intensity in the regions of time t_i where events were observed to occur, conditional on the features x_i. Since there is no constraint on the output of the intensity networks, the first term alone is not useful because λ_θ can be made arbitrarily large during training. The second term in Equation 4 acts as a regularizer on λ_θ. It is a global term (i.e., not specific to any particular event) that penalizes the area under the intensity curve and plays a crucial role in training.

With the goal of scaling up the simulator to large recommendation datasets, the sum over observations in Equation 4 is suggestive of stochastic gradient ascent methods. In more detail, we introduce a stochastic objective $\tilde{\mathcal{L}}(\theta)$ on a sample of a single event at time t_i with features $x(t_i)$,

$$\tilde{\mathcal{L}}(\theta) = \log \lambda_\theta(t_i \mid x(t_i)) - \frac{1}{N}\int_0^T \lambda_\theta(t \mid x(t))\mathrm{d}t, \quad (5)$$

where it can verified that $\mathbb{E}[\tilde{\mathcal{L}}(\theta)] = \lambda_\theta$.

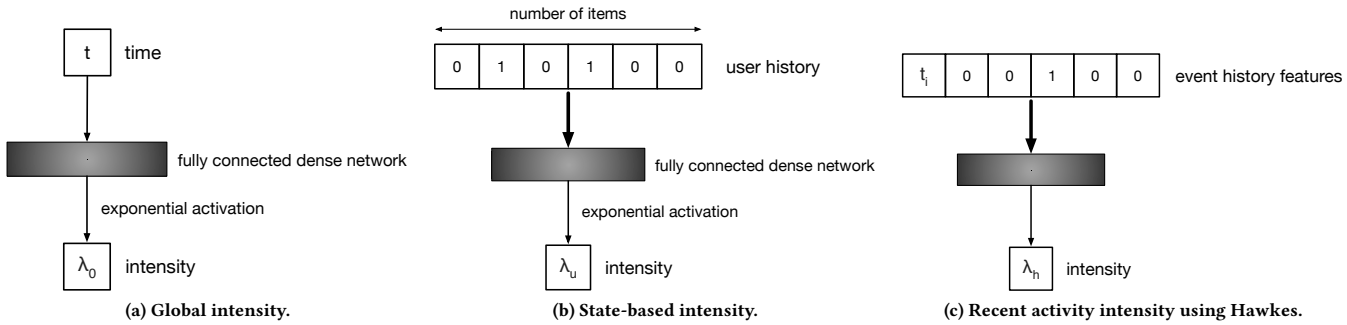
However, Equation 5 is unable to reap the benefits of computationally cheap stochastic gradient updates due to the global regularization term. Specifically, in order to take the gradient of $\tilde{\mathcal{L}}(\theta)$ w.r.t. θ it is necessary to sum over the area of the entire intensity function which involves the set of features x over the entire time range, *including times that were never observed in the dataset.* A uniform sample approximation [17] requires multiple samples and when the cost of featurization is non-trivial it is computationally expensive to repeatedly featurize at unseen time points. Applying importance sample reweighting to account for the shift from data-sampled times to uniform times does not help because the denominator in the collection policy (i.e., data sampling distribution) is the same as the global regularization term. Self-normalized importance sampling [25] results in a constant regularizer that does not even depend on λ. We therefore seek a scalable way to approximate Equation 4 for optimization.

Our solution is to replace the global regularization term in Equation 5 with a single event approximation that is based on treating the area under the intensity curve as a rectangle with width T and height λ at the sample,

$$\hat{\mathcal{L}}(\theta) = \log \lambda_\theta(t_i \mid x(t_i)) - \frac{T}{N}\lambda_\theta(t_i \mid x(t_i)). \quad (6)$$

Note that, Equation 6 is not a Monte Carlo estimate of Equation 4 because the integral in Equation 4 is not over the dataset but over the entire time range. Nonetheless, it is still a useful objective to optimize over. In more detail, we will discuss an intuition for why it works and then present a proof that $\hat{\mathcal{L}}(\theta)$ is a lower bound on $\tilde{\mathcal{L}}(\theta)$ in expectation after appropriate time rescaling. We also provide an empirical study of its utility in training IPP in Section 4.1.

Intuitively, the objective in Equation 6 balances placing high log intensity on the sampled event at time t_i given x_i against the rescaled intensity for the same event. Viewing $k = \frac{N}{T}$ as a *pseudo-count*, Equation 6 is proportional to a log Poisson distribution $\propto \frac{N}{T}\log \lambda_\theta(t_i \mid x_i) - \lambda_\theta(t_i \mid x_i) = \log(\text{Poisson}(k = \frac{N}{T} \mid \lambda_\theta(t_i \mid x_i)))$ up to an additive constant that does not depend on θ. In this way,



(a) Global intensity. (b) State-based intensity. (c) Recent activity intensity using Hawkes.

Figure 4: Architecture for subcomponents of the intensity function. Overall intensity is the sum of outputs of subcomponents.

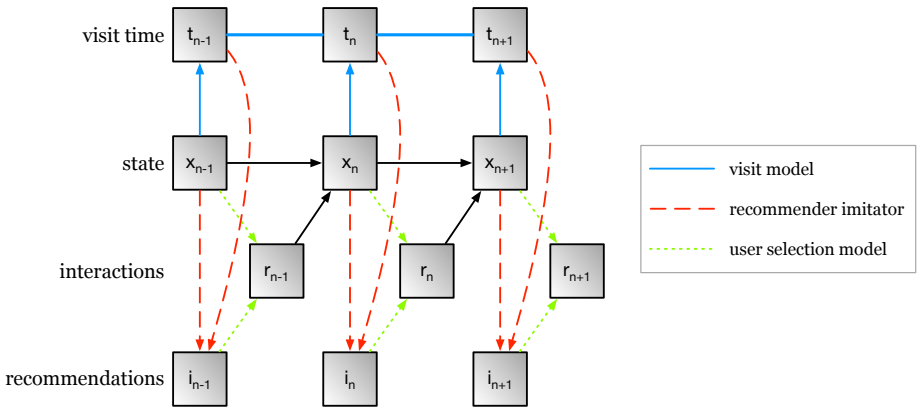


Figure 5: Diagram showing the random variables in the simulator. The trainable functions are the visit model, recommender imitator, and user selection model. The visit times are continuous and inter-connected indicating a non-Markovian assumption. State is updated deterministically given the previous state and interaction.

the stochastic objective $\hat{\mathcal{L}}(\theta)$ induces a set of linked Poisson distributions whose likelihood is to be optimized w.r.t. θ at each event in the training dataset, with pseudo-counts all set to the average number of events per unit of time ($\frac{N}{T}$). Furthermore, it is providential to have the region of λ_θ that is being altered during each stochastic update appear in both the log intensity term and the regularizer term of the gradient because the regularizer locally prevents large increases in the log intensity for the sampled event.

Finally, we prove that $\hat{\mathcal{L}}(\theta)$ from Equation 6 is a lower bound on $\tilde{\mathcal{L}}(\theta)$ from Equation 5 in expectation under the data sampling distribution given the condition that there is on average less than one event per unit of time.

Theorem 3.1. $\mathbb{E}_{N,t\sim\mathcal{D}}[\hat{\mathcal{L}}(\theta)] \leq \mathbb{E}_{N,t\sim\mathcal{D}}[\tilde{\mathcal{L}}(\theta)]$ *when* $T \geq \mathbb{E}[N]$.

Proof. The log intensity term $\log\lambda_\theta(t_i \mid x(t_i))$ is the same for both $\hat{\mathcal{L}}$ and $\tilde{\mathcal{L}}$ and their expectations are identical. Since expectation is a positive linear operator, it only remains to show that the regularizing terms follow the converse inequality. To reduce notation, w.l.o.g. we refer next to the intensity function as $\lambda(t)$ and omit the sampling distribution from the expectation symbol. Starting with condition $T \geq \mathbb{E}[N]$, multiply both sides by $\mathbb{E}\left[\frac{1}{N}\right]$ and manipulate the LHS,

$$T\mathbb{E}\left[\frac{1}{N}\right] = T\mathbb{E}\left[\frac{1}{N}\right]\frac{\mathrm{Var}[N]}{\mathbb{E}[N]} = T\mathbb{E}\left[\frac{1}{N}\right]\mathbb{E}[\lambda(t)] = \mathbb{E}\left[\frac{T}{N}\lambda(t)\right], \tag{7}$$

where we have used the Poisson variance-to-mean ratio and Campbell's theorem. The RHS of the condition is equal to,

$$\mathbb{E}[N]\mathbb{E}\left[\frac{1}{N}\right] = \mathbb{E}\left[\frac{\mathbb{E}[N]}{N}\right] = \mathbb{E}\left[\frac{1}{N}\int_0^T \lambda(t)\mathrm{d}t\right]. \tag{8}$$

Therefore, the event regularizer is an upper bound of the full regularizer in expectation $\mathbb{E}\left[\frac{T}{N}\lambda(t)\right] \geq \mathbb{E}\left[\frac{1}{N}\int_0^T \lambda(t)\mathrm{d}t\right]$ which holds when the expected number of events is less than the total units of time. For model training purposes, we can always rescale the dataset such that this condition holds on a per user basis. □

3.3 Simulation

The goal of simulation is to use a trained model generate a new synthetic dataset consisting of rows $\{(t_n, u_n, i_n, r_n)\}_{n=1}^N$, where t_n is the visit time, u_n is the user ID, i_n is the item ID, and r_n is the outcome of the interaction (e.g., click, purchase, stream). Note that

Algorithm 1: Scalable user-based inhomogeneous Poisson process thinning

Input: Simulator $(\lambda_\theta, m_\theta)$, number of users U, feature function x, max. time T, max. intensity λ_h
Result: Simulated dataset $\mathcal{S} = \{t_n, u_n, i_n, r_n)\}_{n=1}^{N}$

1 Sample number of potential visits $N_u \sim \text{Poisson}(\lambda_h)$ for each user $u \in [1, U]$
2 Sample homogeneous visit times $t_{u,b} \sim \text{Uniform}(0, T)$ for each user $u \in [1, U]$ and potential visit $b \in [1, N_u]$
3 Initialize batch index $b \leftarrow 1$
4 Initialize simulation $\mathcal{S} \leftarrow \emptyset$
5 **while** $b \leq \max_u(N_u)$ **do**
6 **forall** $\{u \mid b \leq N_u\}$ **do**
7 Calculate visit features $\xi \leftarrow x(t_{u,b}, \mathcal{S})$
8 Sample visit acceptance $a \sim \text{Bernoulli}\left(\frac{\lambda(t_{u,b} \mid \xi)}{\lambda_h}\right)$
9 **if** $a == 1$ **then**
10 **while** *visit has more interactions* **do**
11 Sample marking $i, r \sim m(\cdot \mid \xi)$
12 $\mathcal{S} \leftarrow \mathcal{S} \cup (t_{u,b}, u, i, r)$
13 **end**
14 **end**
15 **end**
16 $b \leftarrow b + 1$
17 **end**

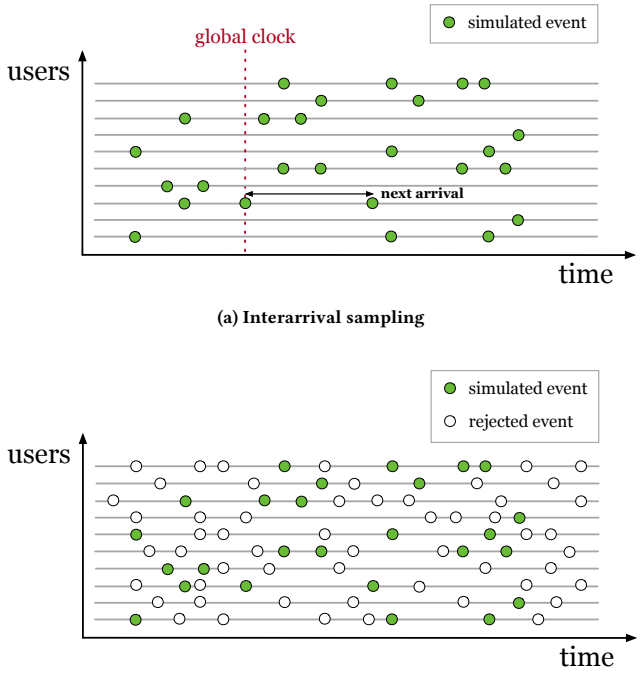


Figure 6: Examples of interarrival time and rejection sampling for IPP.

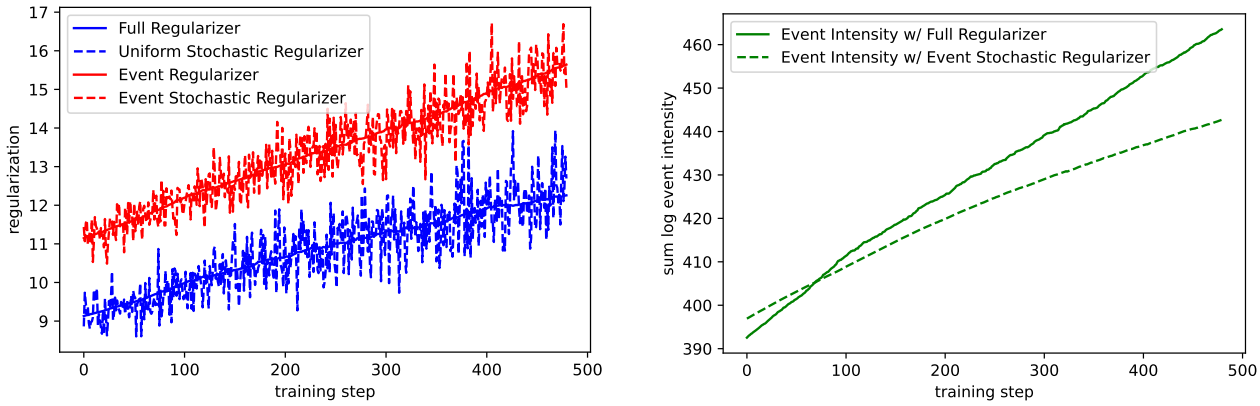
N is a random quantity owing to the fact that visits are randomly generated.

There are two standard approaches to sampling from an inhomogeneous Poisson process: interarrival time sampling and rejection sampling [23]. Interarrival time sampling for IPPs takes the history of events in the sampled trajectory $\{(t_n, u_n, i_n, r_n)\}_{n=1}^{m}$ up to the current time t_m and samples the next visit time and its markings $(t_{m+1}, u_{m+1}, i_{m+1}, r_{m+1})$ and appends it to the history. In practice, inhomogeneous interarrival time sampling requires iteratively advancing a global clock by shorter time steps, many of which do not result in an accepted event . This approach is summarized in Figure 6a. The issue with interarrival time sampling is that it is not known prior to sampling which queries need be made to the simulator models and which features are to be used for each query. As a result, it is not possible to batch feature processing and model prediction to take advantage of multicore processing.

In contrast, full rejection sampling enables planning computations in advance of the sampling and allows scalable simulation. This approach is summarized in Figure 6b. Rejection sampling in IPPs first samples a full set of trajectories from a homogeneous Poisson process with constant intensity $\lambda_h \geq \lambda(t \mid x(t))\ \forall t$ that dominates the IPP intensity λ. This yields a superset of the events that will eventually be the sample from the IPP. Although the state and intensity cannot be determined in advance (as it depends on the outcome of previous interactions that are yet to be sampled) the visit times are known at this step. In the second step, each event m is thinned (i.e. rejected) by flipping a coin with probability of thinning $\frac{\lambda(t_m \mid x(t_m))}{\lambda_h}$. This process can be performed in parallel across all users by iterating over each user's next event asynchronously in simulated time.[3]

Algorithm 1 is the resulting algorithm for scalable user-based simulation. The algorithm takes as inputs the trained simulator consisting of intensity function λ and marking distribution m, as well as the feature function x, maximum time T, and maximum homogeneous Poisson process intensity λ_h. Initially, a set of potential visits is sampled from the homogeneous Poisson process with intensity λ_h; this is achieved by sampling the maximum number of events and choosing their times uniformly. Then, the outer while loop in Line 5 iterates over batches of all active users, defined as those who still have remaining potential visits to process. For each active user, the features are processed given the visit time and history of simulation up to the visit time. Line 8 decides whether the potential visit is accepted as a logged event in the simulation. If so, a set of interactions is generated that depends on the mode of interaction, e.g., the number of interactions for visiting user u could be sampled from a Poisson distribution λ_u that is the empirical average of number of interactions per visit from training data (as we do in Section 4), or could be more structured such as a slate or cascade model [5]. For each interaction, the marking (i, r), consisting of the item impression and outcome of interaction from the recommender imitator and user choice models (respectively), is sampled from the marking distribution m given the features. Sampling each active

[3]That is, the computations are done in parallel but the visit times evaluated in any batch are different (asynchronous).



(a) The event regularizer used to train the IPP is an upper bound in expectation of full regularizer. In practice, stochastic versions of both approaches maintain this relationship.

(b) Sum log intensity of observed events for the intensity trained with full regularizer vs. stochastic event regularizer shows greater tempering for event regularized intensity.

Figure 7: Comparison between approximate scalable training objective and exact objective.

user in batch b is amenable to simple parallelization over multiple cores and/or machines.

4 EMPIRICAL EVALUATION

In this section, we perform experiments with real-world recommendation datasets to evaluate Accordion. Section 4.1 is an empirical study of the likelihood approximation (developed in Section 3.2). In Section 4.2, we consider the task of fitting the exploration hyperparameter on the ContentWise dataset. Finally, in Section 4.3, we evaluate the utility of Accordion in predicting an A/B test of an interactive system where one of the recommenders is changed between test cells while the others are held constant. Overall, this series of empirical studies leads us to the following high-level conclusions: (1) the learnt intensity curves accurately capture user activity, (2) the resulting simulated trajectories exhibit underdispersion artifacts relating to the Poisson process assumptions, and (3) despite this, the key emergent properties of the simulation such as incremental impact on total number of positive interactions provide a valuable signal for navigating model space in an interactive system.

Datasets and Code. Most of our experiments use the ContentWise impressions dataset, a public dataset of interactions logged from a video streaming platform [21]. After removing inactive users and items[4], the dataset comprises 127,904,252 impressions representing the interactions between 15,983 users and 2,211 items across 98 days. The training set takes a random 70% subset of users for the first 70 days of the dataset, the validation dataset the remaining 30% of users over the same time range. The test set spans the last 28 days of the dataset which were unseen during training. The source code to reproduce the results of experiments using ContentWise data is available on GitHub.[5]

Using an observational dataset such as ContentWise is not sufficient for a full evaluation of A/B test prediction because it is biased by the recommenders used during data collection. To address this, we also use a private dataset of an online randomized controlled trial for recommenders where the user is the unit of randomization. The private dataset comes from another video streaming platform and contains a subset of 11,525 users and a subset of 1,723 items over 28 days. The logs for 50% of the users were split into train data and 50% into validation data.

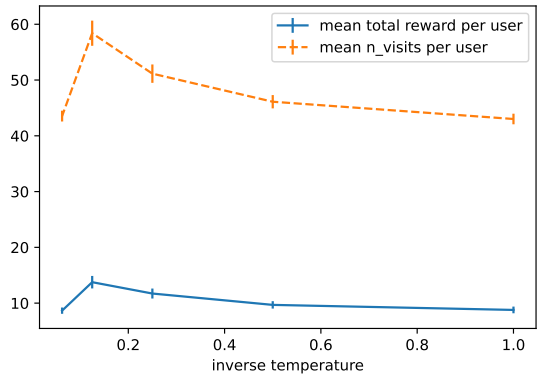
4.1 Poisson Process Likelihood Approximation

Section 3.2 proposes a training objective that is amenable to stochastic gradient ascent on large datasets. In this section, we compare the proposed objective (Equation 6) against the full objective (Equation 4, which does not scale to large datasets) and a uniform time approximation[6] of the full objective [17] (i.e., sampling event times mostly not seen in the dataset) on a subset of data for which all approaches are applicable, i.e., 50 randomly selected users from the ContentWise training data.

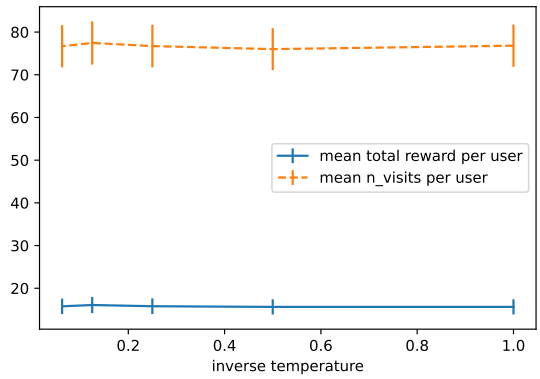
Figure 7a shows the value of the full regularizer, the stochastic regularizer of 100 random events sampled from the dataset (referred to as the *event regularizer*), and the stochastic regularizer of 100 random points sampled uniformly in time for each training step in stochastic gradient ascent. Given the properties discussed in Section 3.2, we expect the full regularizer to tend toward a lower bound of the stochastic regularizer in expectation, and this is what we see in practice throughout training. Further, while the uniform time sampled stochastic regularizer is equal to the full regularizer in expectation, it has slightly higher variance than event regularization due to the fact that it may not sample times with high intensity that contribute the most to the area under the curve. As a result of these properties, the total intensity for observed events, shown in

[4] An inactive user is defined as having strictly fewer than 5 sessions, where a session is defined as any impressions within a half hour period; an inactive item is defined as having strictly fewer than 100 impressions or 5 positive interactions.
[5] https://github.com/jamesmcinerney/accordion

[6] Note that sampling uniformly in time requires featurizing at unseen time points $x(t)$ on each iteration which is more computationally demanding than using the features from observed events.



(a) Hyperparameter sweep using simulations from Accordion. Y-axis shows outcomes of interest in this example (higher is better) and error bars show two times standard error of the mean.



(b) Hyperparameter sweep using simulations where user interactions do not affect visits. Y-axis shows outcomes of interest in this example (higher is better) and error bars show two times standard error of the mean.

Figure 8: Hyperparameter sweep on ContentWise data.

Figure 7b, is more regularized for the event stochastic regularizer than with the full regularizer, helping to reduce overfitting and allowing training to scale up in the number of users.

4.2 Simulation for Hyperparameter Fitting

We next perform experiments to quantify the validity of the simulator on prediction tasks using held-out data. In recognition of the fact that recommender models exist in an interactive system over time, it is common to use exploration in the recommender to provide better data to train on in the next iteration. Options include epsilon-greedy, Boltzmann exploration, Thompson sampling, and upper confidence bounds. Each of these methods has a hyperparameter controlling exploration. Given that changes to the exploration behavior of the algorithm changes the data collected, any offline evaluation that uses a fixed dataset will not take into account the downstream effects of more informative data. On the other hand, simulation can encompass retraining as part of the evolution of the environment over time.

Our experimental setting is as follows. We trained Accordion on our train split of the ContentWise dataset then evaluated a standard recommender (non-negative matrix factorization [7]) on the 200 most active users over 28 days. We added Boltzmann exploration to the recommender scores s to make a policy $h(a_i) \propto s_i^{\frac{1}{M}}$ with the hyperparameter of inverse temperature $\frac{1}{M}$ guiding exploration (i.e., lower inverse temperature has more exploration). The recommender was randomly initialized at the start of the test period, then retrained within the simulation after 3 days, then every 7 days thereafter.[7] Our comparison is against a method that does not vary the pattern of visits in response to interaction outcomes in the simulation, representing methods that take, as given, a set of impressions and adjust only the markings (i.e., items selected and user response). For the invariant visit simulator, we select a random number of visits for each user $N_u \sim \text{Poisson}(28\lambda_u)$ using the empirical average number of visits per day λ_u for user u calculated from the train split, and verified the empirical average by comparing against the real test data, 6.15 visits in test vs. 6.60 visits in train.

Figure 8a shows the result of a sweep over $\frac{1}{M} \in \{2^{-k} \mid k = 0, 1, 2, 3, 4\}$ where $k = 0$ recovers the original recommender without exploration. We found that Accordion shows much greater sensitivity to exploration than the homgeneous approach, shown in Figure 8b. In particular, Figure 8a suggests that $\frac{1}{M} = 2^{-3}$ optimizes the number of visits and consequently positive interactions by balancing exploration with exploitation. The standard errors of the mean in Figure 8b do not validate the selection of any individual inverse temperature. It is not possible to verify these results without collecting a new dataset under the target policy h. We address this shortcoming in the next section with an experimental setting where do verify h online.

4.3 Simulation for Predicting an A/B Test

Finally, we consider the problem of predicting the outcome of an A/B test. A/B tests are an invaluable tool for generating randomized controlled trial data to measure the benefit of a new recommender. The downsides are that they are expensive to perform, have a risk of diminished experience for users, and introduce delay to innovation and development. This motivates debiasing the data offline without online intervention.

In this setting, we assume that a machine learning practitioner has trained a new recommender h offline with the intention of replacing recommender π_1 against a background of several other recommenders, distilled as π_2, that remain unchanged. We refer to the combination (π_1, π_2) as the *control* policy and (h, π_2) as the *target* policy. The intention and hypothesis is that the target policy will increase user satisfaction, as measured, for example, by number of positive interactions or visits during the the test period. See Figure 9 for a diagram of the simulation and test procedure.

To validate the simulator for this task, we train it to imitate an existing interactive system of impressions, visits, and streams

[7] We could have set the training intervals more uniformly but truncated time to first training to reduce number of pure random exploration actions.

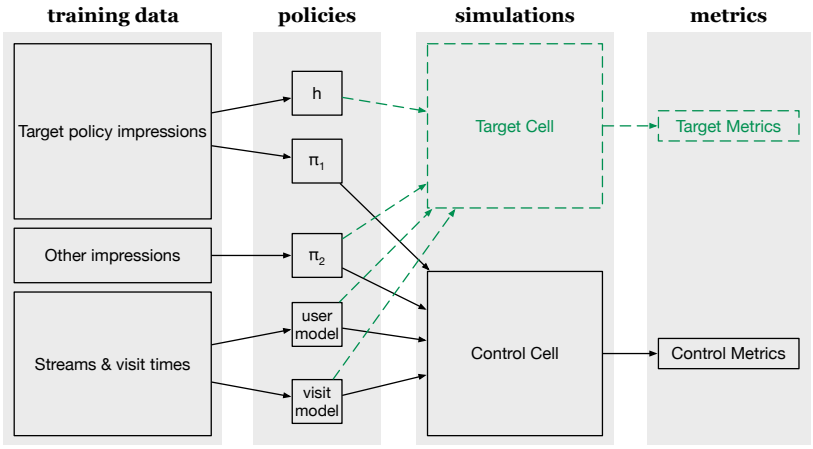
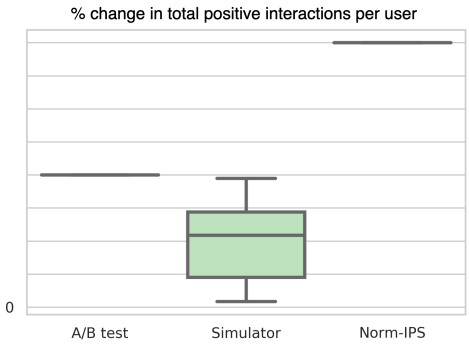
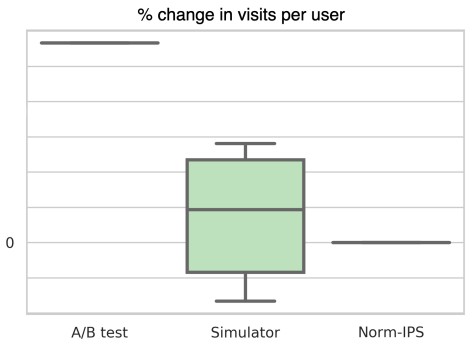


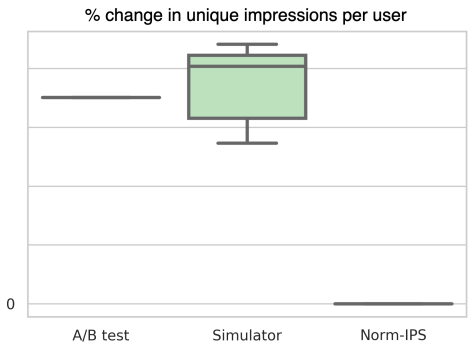
Figure 9: Schematic showing the experimental setup for Section 4.3. Training data are used to learn the policies that are different across A/B test cells (h and π_1) as well as π_2 which remains constant across cells. Models of user choice and visits are also trained. Simulations generate data for the target cell against which the control cell is compared in the final metrics stage.



(a) Change in total positive interactions.



(b) Change in total visits per user.



(c) Change in unique impressions per user.

Figure 10: Results of predicting A/B test results relative to control policy. Error bars show 2 standard deviations. The non-zero values of the y-axis are obscured due to their business sensitive nature and the scale is not constant across figures.

from a private dataset collected from a video streaming platform (as described at the beginning of Section 4), then use the trained simulator to generate new trajectories under the control and target policies. We compare the new trajectories to the existing system by evaluating the number of visits, positive interactions, and impressions.

Figure 10 shows the performance of the simulator on this task against Norm-IPS, a popular existing method for debiasing data offline that uses inverse propensity scores with variance reduction by normalization [25]. Figure 10a shows that the simulator, in estimating the total number of positive interactions, is more conservative and closer to the A/B test result than Norm-IPS. Norm-IPS and related IPS methods (with or without variance reduction) can be over-optimistic because they usually assume that the effects of each logged action are independent. In addition, they provide no signal on how the size of the dataset may change under alternative policies, as shown in Figure 10b and Figure 10c. Simulation captures the effects of interactions on the quantity, timing, and outcome of subsequent interactions and provides valuable predictions about the change in number of visits and impressions. This improvement comes at the cost of introducing bias into the estimates, both finite data and in the limit, as a result of the modeling assumptions. In addition, optimizing a lower bound to the objective (Equation 6) causes the method to underestimate the number of events. However, as observed in Figure 10, an asymptotically unbiased method also becomes biased when its strong assumptions are violated.

5 CONCLUSIONS & FUTURE WORK

In this paper, we presented a framework for simulating interactive systems with multiple recommenders for realistic and interpretable insights. We introduced Accordion, a trainable long-term simulator for variable-sized datasets and applied it to a public and private dataset on two important tasks. The training objective is scalable to many users and features and is a lower bound of the true objective. In future work, we aim to improve calibration in the estimated intensity function to account for the lower bound and to extend the Poisson model to allow overdispersion in the number of events. Furthermore, we aim explore the wide range of intensity functions that can be brought to bear on visit patterns such as weekly cadence, alternative Hawkes decay, and featurizing items in the intensity function to generalize to new items.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers and our Netflix colleagues for all their helpful questions and feedback.

REFERENCES

[1] Norman R. Baker and Richard E. Nance. 1968. The use of simulation in studying information storage and retrieval systems. *American Documentation* 19, 4 (1968), 363–370. https://doi.org/10.1002/asi.5090190402

[2] Christopher M Bishop. 2006. *Pattern recognition and machine learning*. Springer.

[3] Alex Boyd, Robert Bamler, Stephan Mandt, and Padhraic Smyth. 2020. User-dependent neural sequence models for continuous-time event data. In *Advances in Neural Information Processing Systems*, Vol. 33. 21488–21499.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *arXiv 1606.01540* (2016). arXiv:1606.01540 http://arxiv.org/abs/1606.01540

[5] Ben Carterette, Evangelos Kanoulas, and Emine Yilmaz. 2011. Simulating simple user behavior for system effectiveness evaluation. In *International Conference on Information and Knowledge Management, Proceedings*. 611–620. https://doi.org/10.1145/2063576.2063668

[6] Allison J B Chaney, Brandon M Stewart, and Barbara E Engelhardt. 2018. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, New York, NY, USA. https://doi.org/10.1145/3240323.3240370

[7] Andrzej Cichocki and Anh Huy Phan. 2009. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* E92-A, 3 (2009), 708–721. https://doi.org/10.1587/transfun.E92.A.708

[8] Michael D Cooper. 1973. A simulation model of an information retrieval system. In *Inform Stor. Retr*, Vol. 9. Pergamon Press, 13–32. https://www.sciencedirect.com/science/article/pii/0020027173900041

[9] Bradley Efron and Trevor Hastie. 2016. *Computer age statistical inference*. Vol. 5. Cambridge University Press.

[10] Jin Huang, Harrie Oosterhuis, Maarten De Rijke, and Herke Van Hoof. 2020. Keeping dataset biases out of the simulation: a debiased simulator for reinforcement learning based recommender systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*, Vol. 22. Virtual Event. https://doi.org/10.1145/3383313.3412252

[11] Eugene Ie, Chih-wei Hsu, Martin Mladenov, Vihan Jain, Sanmit Narvekar, Jing Wang, Rui Wu, and Craig Boutilier. 2019. RecSim: s configurable simulation platform for recommender systems. *arXiv 1909.04847* (2019). arXiv:1909.04847 http://arxiv.org/abs/1909.04847

[12] J.F.C. Kingman. 2005. Poisson processes. In *Encyclopedia of Biostatistics*. John Wiley & Sons, Ltd, Chichester, UK. https://doi.org/10.1002/0470011815.b2a07042

[13] Benjamin Letham, Lydia M. Letham, and Cynthia Rudin. 2016. Bayesian inference of arrival rate and substitution behavior from sales transaction data with stockouts. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, 1695–1704. https://doi.org/10.1145/2939672.2939810 arXiv:1502.04243

[14] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *The Web Conference 2018 - Proceedings of the World Wide Web Conference, WWW 2018*. Association for Computing Machinery, Inc, 689–698. https://doi.org/10.1145/3178876.3186150 arXiv:1802.05814

[15] Jean Michel Marin, Pierre Pudlo, Christian P. Robert, and Robin J. Ryder. 2012. Approximate Bayesian computational methods. *Statistics and Computing* 22, 6 (2012), 1167–1180. https://doi.org/10.1007/s11222-011-9288-2 arXiv:1101.0955

[16] James McInerney, Benjamin Lacker, Samantha Hansen, Karl Higley, Hugues Bouchard, Alois Gruson, and Rishabh Mehrotra. 2018. Explore, exploit, and explain: personalizing explainable recommendations with bandits. *RecSys 2018 - 12th ACM Conference on Recommender Systems* (2018), 31–39. https://doi.org/10.1145/3240323.3240354

[17] Hongyuan Mei and Jason Eisner. 2017. The neural Hawkes process: a neurally self-modulating multivariate point process. In *arXiv 1612.09328*.

[18] Martin Mladenov, Chih-Wei Hsu, Vihan Jain, Eugene Ie, Christopher Colby, Nicolas Mayoraz, Hubert Pham, Dustin Tran, Ivan Vendrov, and Craig Boutilier. 2021. RecSim NG: toward principled uncertainty modeling for recommender ecosystems. In *arXiv 2103.08057*. arXiv:2103.08057 http://arxiv.org/abs/2103.08057

[19] Javed Mostafa, Snehasis Mukhopadhyay, and Mathew Palakal. 2003. Simulation studies of different dimensions of users' interests and their impact on user modeling and information filtering. *Information Retrieval* 6, 2 (2003), 199–223. https://doi.org/10.1023/A:1023932221048

[20] Tin Lok James Ng and Andrew Zammit-Mangion. 2020. Non-homogeneous Poisson process intensity modeling and estimation using measure transport. *arXiv 2007.00248* (2020).

[21] Fernando B. Pérez Maurera, Maurizio Ferrari Dacrema, Lorenzo Saule, Mario Scriminaci, and Paolo Cremonesi. 2020. ContentWise impressions: an industrial dataset with impressions included. In *International Conference on Information and Knowledge Management, Proceedings*. ACM, New York, NY, USA, 3093–3100. https://doi.org/10.1145/3340531.3412774

[22] Oliver Ratmann, Ole Jørgensen, Trevor Hinkley, Michael Stumpf, Sylvia Richardson, and Carsten Wiuf. 2007. Using likelihood-free inference to compare evolutionary dynamics of the protein networks of H. pylori and P. falciparum. *PLoS Computational Biology* 3, 11 (2007), 2266–2278. https://doi.org/10.1371/journal.pcbi.0030230

[23] Marian Andrei Rizoiu, Young Lee, Swapnil Mishra, and Lexing Xie. 2017. A tutorial on hawkes processes for events in social media. In *arXiv 1708.06401*. arXiv. arXiv:1708.06401

[24] Torbjörn Sjöstrand, Stephen Mrenna, and Peter Skands. 2008. A brief introduction to PYTHIA 8.1. *Computer Physics Communications* 178, 11 (2008), 852–867. https://doi.org/10.1016/j.cpc.2008.01.036 arXiv:0710.3820

[25] Adith Swaminathan and Thorsten Joachims. 2015. The Self-Normalized Estimator for Counterfactual Learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'15)*. MIT Press, Cambridge, MA, USA, 3231–3239.

[26] Flavian Vasile Criteo, David Rohde, Stephen Bonner, Travis Dunlop, Flavian Vasile, and Alexandros Karatzoglou. 2018. RecoGym: A reinforcement learning environment for the problem of product recommendation in online advertising. In *RecSys*, Vol. 18. arXiv:1808.00720v2 https://github.com/criteo-research/reco-gym

[27] Simon N. Wood. 2010. Statistical inference for noisy nonlinear ecological dynamic systems. *Nature* 466, 7310 (2010), 1102–1104. https://doi.org/10.1038/nature09319