Modeling the Dynamics of Learning Activity on the Web

Charalampos Mavroforakis[*]
Boston University
cmav@bu.edu

Isabel Valera
MPI-SWS
ivalera@mpi-sws.org

Manuel Gomez Rodriguez
MPI-SWS
manuelgr@mpi-sws.org

ABSTRACT

People are increasingly relying on social media and the Web to find solutions to their problems in a wide range of domains. In this setting, closely related problems often lead to the same characteristic *learning pattern*—people sharing a similar problem visit closely related pieces of information, perform almost identical queries or, more generally, take a series of similar *actions* at a similar pace. In this paper, we introduce a novel modeling framework for clustering *continuous-time* grouped streaming data, the Hierarchical Dirichlet Hawkes process (HDHP), which allows us to automatically uncover a wide variety of learning patterns from detailed traces of learning activity. Our model allows for efficient inference, scaling to millions of actions and thousands of users. Experiments on real data from Stack Overflow reveal that our framework recovers meaningful learning patterns, accurately tracks users' interests and goals over time and achieves better predictive performance than the state of the art.

Keywords

learning activity modeling; user interest tracking; continuous-time data clustering.

1. INTRODUCTION

Learning is nowadays becoming an online activity – people routinely use a wide variety of *learning platforms* on the Web, ranging from wikis and question answering (Q&A) sites to online communities and blogs, to learn about a large range of topics. In this context, people find solutions to their problems by looking at closely related pieces of information, executing a sequence of queries or, more generally, performing a series of *learning actions*. For example, a software engineer may read several answers within a Q&A site to quickly solve a specific programming problem; a high school student may study several closely related wiki pages over a week to prepare an essay about a historical event; and, a machine learning researcher may monthly check a specialized blog written by one of her peers to learn about a new machine learning technique. In all these examples, people are performing a series of learning actions characterized by their content (*i.e.*, programming, history, or machine learning) and time duration (*i.e.*, a few hours, days, or months) to solve a predefined learning task (*i.e.*, solving a programming problem, writing an essay, or learning about a machine learning method). In this context, one expects that people presented with similar problems will perform *similar* sequences of actions, both in terms of content and timing. Consequently, one may think of a sequence of related actions as a *task* and of *similar* tasks as realizations of a unique *learning pattern*, which characterizes both the content and timing of the tasks.

In this work, we introduce the Hierarchical Dirichlet Hawkes Process (HDHP), a novel probabilistic model for clustering *continuous-time* grouped streaming data, and use it to uncover the above mentioned learning patterns in online learning platforms. The HDHP leverages the properties of two other processes: (i) the Hierarchical Dirichlet Process (HDP) [19] and (ii) the Hawkes process [13]. The former is a popular Bayesian nonparametric model consisting of a hierarchy of Dirichlet processes (DP) [12], and is commonly used for solving clustering problems involving multiple groups of data. The latter is a temporal point process particularly well fitted to model social activity [11, 20, 22]. Here, the HDP is used to account for an infinite number of learning patterns, which are shared across users (groups) of an online learning platform, while the Hawkes is used to characterize the temporal dynamics of each learning pattern. More specifically, the HDHP models each user's learning activity as a multivariate Hawkes process, with as many dimensions as the number of learning patterns (*i.e.*, infinite). The parameters of this process are shared across all the users and sampled, together with the parameters of the associated content, from a DP. Every time that a user decides to perform a new action, she may opt for starting a new task (*i.e.*, a new realization of a learning pattern) or follow-up on one of her ongoing ones.

We develop an efficient inference algorithm for our model, based on the sequential Monte Carlo [16], which scales to millions of actions and thousands of users. We apply our algorithm on real-world data from Stack Overflow, using $\sim$1.6 million questions performed by 16,000 users over a four year period. Our results show that our model can recover meaningful learning patterns, it can accurately track users' interests and goals over time and, by leveraging both content and temporal information, it can provide better

[*] This work was done during this author's internship at MPI-SWS.

WWW 2017, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4913-0/17/04.
http://dx.doi.org/10.1145/3038912.3052669

predictive performance compared to the HDP [19]. A Python implementation of the proposed HDHP is available online[1].

Related work. The Hierarchical Dirichlet Hawkes process (HDHP) can be viewed as a model for clustering grouped continuous-time streaming data. In our application domain, each group of data corresponds to a user's online actions and the clusters correspond to learning patterns, shared across all the users. Therefore, our work relates, on the one hand, to models for clustering groups of data [6, 18], and, on the other hand, to models for clustering (ungrouped) streaming data [3, 5, 2, 10]. To the best of our knowledge, models for clustering grouped streaming data are nonexistent to date.

The most popular models for clustering groups of data originate from the topic modeling literature: the Latent Dirichlet Allocation (LDA) [6], in which the number of clusters is predefined, and the HDP [19], LDA's nonparametric counterpart. There, each document is transformed into a *bag-of-words* and is modeled as a mixture of topics that are shared across all documents. More generally, models for clustering groups of data typically consider that each observation (word) within a group (document) is a sample from a mixture model, and the mixture components are shared among groups. One can use the HDP to cluster users' activity on the Web, however observations are assumed to be exchangeable and thus it cannot account for the temporal dynamics of learning activity. As a consequence, it is unable to track users' interests and goals over time and provides a worse fit to the data, as shown in Section 4.

Models for clustering streaming data can incorporate temporal dynamics [2, 3], however they can only handle a single stream of data and thus cannot be used to jointly model several users' learning activity. Additionally, most of these models discretize the time dimension into bins [2, 3], introducing additional tuning parameters, and ignoring the self-excitation across actions [5], a phenomenon regularly observed in social activity data [11]. Perhaps the most closely related work to ours is the recently proposed Dirichlet Hawkes Process (DHP) [10], a continuous-time model for streaming data that allows for self-excitation. However, DHP suffers from a significant limitation: the lack of an underlying DP (or, in fact, any other Bayesian nonparametric) prior on the cluster distribution compromises the identifiability and reproducibility of the model. Additionally, from the perspective of our application, DHP only allows for a single data stream (a single user) and enforces clusters to be forgotten after some time. The latter is an overly restrictive assumption, since a user may perform similar actions, *i.e.*, belonging to the same learning pattern, over widely spaced intervals of time. As a consequence, the DHP is not suitable to model learning activity data on the Web, and therefore, to track users' interest over time.

2. PRELIMINARIES

In this section, we briefly review the major building blocks of the Hierarchical Dirichlet Hawkes process (HDHP): the Hierachical Dirichlet process (HDP) [19] and the Hawkes process [1].

2.1 Hierarchical Dirichlet Process

The HDP is a Bayesian nonparametrics prior, useful for clustering grouped data [19], which allows for an unbounded number of clusters whose parameters are shared across all the groups. It has been broadly applied for topic modeling as the nonparametric counterpart of the Latent Dirichlet Allocation (LDA), where the number of topics is finite and predefined. More specifically, this process defines a hierarchy of Dirichlet processes (DPs), in which a set of random probability measures $G_j \sim DP(\beta_1, G_0)$ (one for each group of data) are distributed as DPs with concentration parameter β_1 and base distribution G_0. The latter is also distributed as a DP, *i.e.*, $G_0 \sim DP(\beta_0, H)$. In the HDP, the distributions G_j share the same support as G_0, and are conditionally independent given G_0.

Chinese Restaurant Franchise. An alternative representation of the HDP is the Chinese Restaurant Franchise Process (CRFP), which allows us not only to obtain samples from the HDP but also to derive efficient inference algorithms. The CRFP assumes a franchise with as many restaurants as the groups of data (*e.g.*, number of documents), where all of the restaurants share the same menu with an infinite number of dishes (or clusters). In particular, one can obtain samples from the HDP as follows:

1. Initialize the total number of dishes $L = 0$, and the total number of tables in each restaurant $K_r = 0$, for $r = 1, \dots, R$, with R being the total number of restaurants in the franchise.
2. For each restaurant $r = 1, \dots, R$:
   For customer $i = 1, \dots, N_r$ (N_r is the total number of customers entering restaurant r):
   - Sample the table for the i-th customer in restaurant r from a multinomial distribution with probabilities
   $$\begin{aligned} \Pr(b_{ri} = k) &= \tfrac{n_{rk}}{\beta_1 + i - 1} \quad \text{for} \quad k = 1, \dots, K_r \\ \Pr(b_{ri} = K_r + 1) &= \tfrac{\beta_1}{\beta_1 + i - 1} \end{aligned} \qquad (1)$$
   where $n_{rk} = \sum_{j=1}^{i-1} \mathbb{I}(b_{rj} = k)$ is the number of customers seated at the k-th table of the r-th restaurant.
   - If $b_{ri} = K_r + 1$, *i.e.*, the i-th customer sits at a new table, sample its dish from a multinomial distribution with probabilities
   $$\begin{aligned} \Pr(\phi_{r(K_r+1)} = \varphi_\ell) &= \tfrac{m_\ell}{K + \beta_0} \quad \text{for } \ell = 1, \dots, L \\ \Pr(\phi_{r(K_r+1)} = \varphi_{L+1}) &= \tfrac{\beta_0}{K + \beta_0} \end{aligned} \qquad (2)$$
   where $K = \sum_{j=1}^{r} K_j$ is the total number of tables in the franchise, $m_\ell = \sum_{j=1}^{r} \sum_{k=1}^{K_j} \mathbb{I}(\phi_{jk} = \varphi_\ell)$ is the total number of tables serving dish φ_ℓ in the franchise, and $\varphi_{L+1} \sim H(\varphi)$ is the new dish, *i.e.*, the parameters of the new cluster.
   - Increase the number of tables in the r-th restaurant $K_r = K_r + 1$ and, if $\phi_{rK_r} = \varphi_{L+1}$ (*i.e.*, the new table is assigned to a new dish/cluster), increase also the total number of clusters in the franchise $L = L + 1$.

Note that, although in the process above we have generated the data (customers) for each group (restaurant) sequentially, due to the exchangeability properties of the HDP, the resulting distribution of the data is invariant to the order at which customers are assumed to enter any of the restaurants [19].

2.2 Hawkes Process

A Hawkes process is a stochastic process in the family of temporal point processes [1], whose realizations consist of lists of discrete actions (often called events) localized in time, $\{t_1, t_2, \dots, t_n\}$ with $t_i \in \mathbb{R}^+$. A temporal point process can be equivalently represented as a counting process, $N(t)$,

[1] https://github.com/Networks-Learning/hdhp.py

which records the number of actions before time t. The probability of an action happening in a small time window $[t, t + dt)$ is given by $\Pr(dN(t) = 1|\mathcal{H}(t)) = \lambda^*(t)dt$, where $dN(t) \in \{0, 1\}$ denotes the increment of the process, $\mathcal{H}(t)$ denotes the history of actions up to but not including time t, $\lambda^*(t)$ is the conditional intensity function (intensity, in short), and the sign * indicates that the intensity may depend on the history $\mathcal{H}(t)$. In the case of Hawkes processes, the intensity function adopts the following form:

$$\lambda^*(t) = \mu + \sum_{t_i \in \mathcal{H}(t)} \kappa_\alpha(t, t_i), \tag{3}$$

where μ is the base intensity and $\kappa_\alpha(t, t_i)$ is the triggering kernel, which is parametrized by α. Note that this intensity captures the self-excitation phenomenon across actions and thus allows modeling bursts of activity. As a consequence, the Hawkes process has been increasingly used to model social activity [11, 20, 22], which is characterized by bursts of rapidly occurring actions separated by long periods of inactivity [4]. Finally, given the history of actions in an observation window $[0, T)$, denoted by $\mathcal{H}(T)$, we can express the log-likelihood of the observed data as

$$\mathfrak{L}_T = \sum_{i:t_i \in \mathcal{H}(T)} \log \lambda^*(t_i) - \int_0^T \lambda^*(\tau)\, d\tau. \tag{4}$$

3. LEARNING ACTIVITY MODEL

In an online learning platform, users find solutions to their problems by sequentially looking for closely related pieces of information within the site, executing a sequence of queries or, more generally, performing a series of online *actions*. In this context, one may expect people addressing similar problems to undertake similar sequences of actions, which in turn can be viewed as realizations of an unbounded number of *learning patterns*. Here, we assume that each action is linked to some particular content and we propose a modeling framework that characterizes sequences of actions by means of their timestamps as well as their associated content of these actions. Next, we formulate our model for online learning activity, starting by describing the data it is designed for.

Learning activity data and generative process. Given an online learning platform with a set of users $\mathcal{U}$, we represent the *learning actions* of each user as a triplet

$$e := (\overset{\text{time}}{\overset{\downarrow}{t}}\ ,\ \underset{\text{content}}{\underset{\uparrow}{\boldsymbol{\omega}}}\ ,\ \overset{\text{learning pattern}}{\overset{\downarrow}{p}}\ ), \tag{5}$$

which means that at time t the user took an action linked to content $\boldsymbol{\omega}$ and this action is associated to the learning pattern p, which is hidden. Then, we denote the history of learning actions taken by each user u up to, but not including, time t as $\mathcal{H}_u(t)$.

We represent the times of each user u's learning actions within the platform as a set of counting processes, $\mathbf{N}_u(t)$, in which the ℓ-th entry counts the number of times up to time t that user u took an action associated to the learning pattern ℓ. Then, we characterize these counting processes using their corresponding intensities as $\mathbb{E}[d\mathbf{N}_u(t)|\mathcal{H}_u(t)] = \boldsymbol{\lambda}^*_u(t)\, dt$, where $d\mathbf{N}_u(t) = [dN_{u,\ell}(t)]_{\ell \in [L]}$ denotes the number of learning actions in the time window $[t, t + dt)$ for each learning pattern, $\boldsymbol{\lambda}^*_u(t) = [\lambda^*_{u,\ell}(t)]_{\ell\in[L]}$ denotes the corresponding pattern intensities, L is the number of learning patterns, and the sign * indicates that the intensities may depend on the user's history, $\mathcal{H}_u(t)$. Additionally, for each learning action $e = (t, \boldsymbol{\omega}, p)$, the content $\boldsymbol{\omega}$ is sampled from a distribution $f(\boldsymbol{\omega}|p)$, which depends on the corresponding learning pattern p. Here, in order to account for an unbounded number of learning patterns, *i.e.*, $L \to \infty$, we assume that the learning pattern distribution follows a Dirichlet process (DP). Next, we specify the functional form of the user intensity associated to each learning pattern and the content distribution, and we elaborate further on the learning pattern distribution.

Intensity of the user learning activity. Every time user u performs a learning action, she may opt to either start a new *task*, defined as a sequence of learning actions similar in content and performed closely in time (*i.e.*, a realization of a learning pattern), or to follow-up on an already ongoing task. The multivariate Hawkes process [1], described in Section 2.2, presents itself as a natural choice to model this behavior. This way, each dimension ℓ corresponds to a learning pattern ℓ and its associated intensity is given by

$$\lambda^*_{u,\ell}(t) = \underbrace{\mu_u\ \pi_\ell}_{\text{new task}} + \overbrace{\sum_{j:t_j\in\mathcal{H}_u(t),\ p_j=\ell} \kappa_\ell(t, t_j)}^{\text{follow-up}}. \tag{6}$$

Here, the parameter $\mu_u \geq 0$ accounts for the rate at which user u starts new tasks, $\pi_\ell \in [0, 1]$ is the probability that a user adopts learning pattern ℓ (referred to as *learning pattern popularity* from now on), and $\kappa_\ell(t, t')$ is a nonnegative kernel function that models the decaying influence of past events in the pattern's intensity. Similarly to several works on modeling users' online activity [14, 20, 21], here we opt for an exponential kernel function in the form $\kappa_\ell(t, t') = \alpha_\ell \exp(-\nu(t - t'))$, where α_ℓ controls the self-excitation (or burstiness) of the Hawkes process and ν controls the decay.

Content distribution. We gather the content associated to each learning action $e = (t, \boldsymbol{\omega}, p)$ as a vector $\boldsymbol{\omega}$, in which each element is a word sampled from a vocabulary $\mathcal{W}$ as

$$\omega_j \sim Multinomial(\boldsymbol{\theta}_p), \tag{7}$$

where $\boldsymbol{\theta}_p$ is a $|\mathcal{W}|$-length vector indicating the probability of each word to appear in content from pattern p.

Learning pattern parameters. The distribution of the learning patterns is sampled from a DP, $G_0 \sim DP(\beta, H)$, which can be alternatively written as

$$G_0 = \sum_{\ell=1}^{\infty} \pi_\ell \delta_{\varphi_\ell}, \tag{8}$$

where $\boldsymbol{\pi} = (\pi_\ell)_{\ell=1}^{L=\infty} \sim GEM(\beta)$ is sampled from a stick breaking process [15] and $\varphi_\ell = \{\alpha_\ell, \boldsymbol{\theta}_\ell\} \sim H(\varphi)$.

Remarks. Overall, the proposed learning activity model, which we refer to as the Hierarchical Dirichlet Hawkes process (HDHP), is based on a 2-layer hierarchical design, which is illustrated in Figure 1. The top layer is a Dirichlet process that determines the learning pattern distribution, and the bottom layer corresponds to a collection of independent multivariate Hawkes processes (while in the HDP [19] it corresponds to a collection of DPs), one per user, with as many dimensions as the number of learning patterns, *i.e.*, infinite. In the HDHP, the popularity of each learning pattern, or equivalently the probability of assigning a new task to it, is constant over time and given by the distribution G_0. However, the probability distribution of the learning patterns for each specific user evolves continuously over time and directly depends on her instantaneous intensity. Finally, we remark that, due to the infinite dimensionality of the Hawkes process that captures the learning activity of each user, sampling or performing inference directly on this model is intractable. Fortunately, we can benefit from the properties of both the Hawkes and the DP, and propose an alternative generative process that we can then utilize to efficiently obtain samples of, as well as infer, the HDHP.

3.1 Tractable model representation

Similarly to the HDP, we can generate samples from the proposed HDHP by following a generative process similar to the CRFP. To this end, we leverage the properties of the Hawkes process and represent the learning actions of all the users in the learning platform as a multivariate Hawkes process, with as many dimensions as the users, from which we sample the user and the timestamp associated to the next learning action. This action is then assigned to either an existing or a new task with a probability that depends on the history of that user up to, but not including, the time of the action. When initiating a new task, the associated learning pattern is sampled from a distribution that accounts for the overall popularity of all the learning patterns. We finally sample the action content $\boldsymbol{\omega}$ as we discussed previously.

In the process described above, each user can be viewed as a restaurant, each action as a customer, each task as a table, and each pattern as a dish, as in the original CRFP. Hence, if we assume a set of users $\mathcal{U}$ and vocabulary $\mathcal{W}$ for the content, we can generate N learning actions as follows:

1. Initialize the *total number of tasks* $K = 0$ and the *total number of learning patterns* $L = 0$.
2. For $n = 1, \ldots, N$:
   (a) Sample the time t_n and user $u_n \in \mathcal{U}$ for the new action, such that $t_n > t_{n-1}$, as in [20]

$$(t_n, u_n) \sim \text{Hawkes}\begin{pmatrix} \mu_1 + \sum_{i=1}^{n-1} \kappa_{b_i}(t_n, t_i)\mathbb{I}(u_i = 1) \\ \vdots \\ \mu_{\mathcal{U}} + \sum_{i=1}^{n-1} \kappa_{b_i}(t_n, t_i)\mathbb{I}(u_i = \mathcal{U}) \end{pmatrix} \tag{9}$$

   (b) Sample the task b_n for the new action from a multinomial distribution with probabilities

$$\begin{aligned} \Pr(b_n = k) &= \tfrac{\lambda^*_{u_n,k}(t_n)}{\lambda^*_{u_n}(t_n)}, \quad \text{for} \quad k = 1, \ldots, K \\ \Pr(b_n = K+1) &= \tfrac{\mu_{u_n}}{\lambda^*_{u_n}(t_n)} \end{aligned} \tag{10}$$

   where $\lambda^*_{u_n,k}(t_n) = \sum_{i=1}^{n-1} \kappa_{b_i}(t_n, t_i)\mathbb{I}(u_i = u_n, b_i = k)$, and $\lambda^*_{u_n}(t_n) = \mu_{u_n} + \sum_{i=1}^{n-1} \kappa_{b_i}(t_n, t_i)\mathbb{I}(u_i = u_n)$ is the total intensity of user u_n at time t_n.

   (c) If $b_n = K + 1$, assign the new task to a learning pattern with probability

$$\begin{aligned} \Pr(\phi_{K+1} = \varphi_\ell) &= \tfrac{m_\ell}{K+\beta}, \quad \text{for } \ell = 1, \ldots, L \\ \Pr(\phi_{K+1} = \varphi_{L+1}) &= \tfrac{\beta}{K+\beta} \end{aligned} \tag{11}$$

   where $m_\ell = \sum_{k=1}^{K} \mathbb{I}(\phi_k = \varphi_\ell)$ is the number of tasks assigned to learning pattern ℓ across all users, and $\varphi_{L+1} = \{\alpha_{L+1}, \boldsymbol{\theta}_{L+1}\}$ is the set of parameters of the new learning pattern $L + 1$, which we sample from $\alpha_{L+1} \sim Gamma(\tau_1, \tau_2)$ and $\boldsymbol{\theta}_{L+1} \sim Dirichlet(\eta_0)$. Then, increase the number of tasks $K = K + 1$ and, if $\phi_{K+1} = \varphi_{L+1}$, increase also the number of clusters $L = L + 1$.

   (d) Sample each word in the content $\boldsymbol{\omega}_n$ from $\omega_{n,j} \sim Multinomial(\boldsymbol{\theta}_{b_n})$.

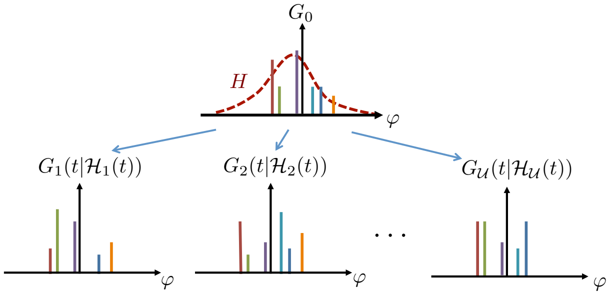
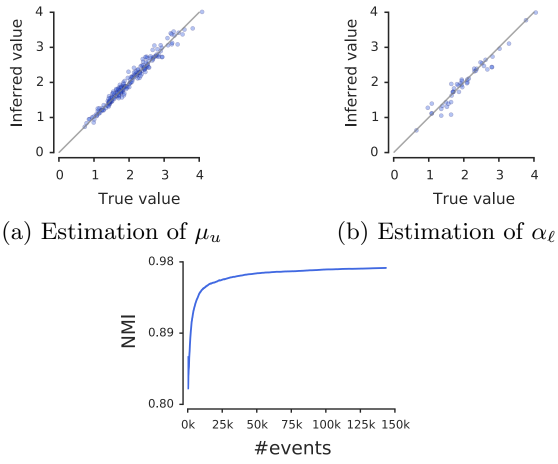

G_0
H
φ
$G_1(t|\mathcal{H}_1(t))$
$G_2(t|\mathcal{H}_2(t))$
$G_{\mathcal{U}}(t|\mathcal{H}_{\mathcal{U}}(t))$
$\cdots$
φ
φ
φ

Figure 1: Illustration of the Hierarchical Dirichlet Hawkes process. The top layer of the hierarchy corresponds to a Dirichlet process with base intensity H and concentration parameter β, *i.e.*, $G_0 \sim DP(\beta, H)$, which determines both the popularity and the parameters of an unbounded number of learning patterns. The bottom layer corresponds to as many multivariate Hawkes processes as the groups of data, with one dimension per learning pattern, whose parameters are given by the DP in the top layer. The popularity (or probability) of each learning pattern is constant over time and given by the distribution G_0, however, the probability distribution of the learning patterns for each user evolves continuously over time and depends on the instantaneous user intensity.

Remarks. Note that, in the process above, both users and learning patterns are exchangeable. However, contrary to the CRFP, the generated data consist of a sequence of discrete events localized in time, which therefore do not satisfy the exchangeability property. As a consequence, this generative process, and therefore, its complexity, differs from the standard CRFP only in two steps. First, it needs to sample the event time and user from a Hawkes process as in Eq. 9, which can be done in linear time with respect to the number of users [11]. Second, while the CRFP only accounts for the number of customers at each table, the above process needs to evaluate the intensity associated with each table (see Eq. 10), which can be updated in O(1) using the properties of the exponential function.

We also stress that although the above generative process resembles the Dirichlet Hawkes process (DHP) [10], they differ in two key factors. First, the DHP can only generate a single sequence of events, while the above process can generate an independent sequence for each user. Second, the DHP does not instantiate an explicit prior distribution on the clusters, which results in a lack of identifiability and reproducibility of the model. In other words, new events in the DHP are only allowed to join a new or a currently active cluster—once a cluster "dies" (*i.e.*, its intensity becomes negligible), no new event can be assigned to it anymore. As a result, two bursts of events that are similar in content and dynamics but widely separated in time will be assigned to different clusters, leading to multiple copies of the same cluster. In contrast, our generative process ensures the identifiability and reproducibility of the model by placing a DP prior on the learning pattern (cluster) distribution, and using the CRFP to integrate out the learning pattern popularity.



(a) Estimation of μ_u (b) Estimation of α_ℓ

(c) Clustering accuracy

Figure 2: Evaluation of the inference algorithm at recovering the model parameters and the latent learning pattern assigned to each event on 150k synthetically generated data.

3.2 Inference

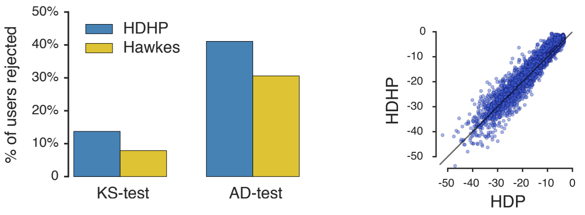
Given a collection of N observed learning actions performed by a set of users $\mathcal{U}$ during a time period $[0, T)$, our goal is to infer the learning patterns that these actions belong to. To efficiently sample from the posterior distribution, similarly to [10], we leverage the generative process described in Section 3.1. We then derive a sequential Monte Carlo (SMC) algorithm that exploits the temporal dependencies in the observed data to sequentially sample the latent variables associated with each learning action. In particular, the posterior distribution $p(b_{1:N}|t_{1:N}, u_{1:N}, \boldsymbol{\omega}_{1:N})$ is sequentially approximated with a set of $\mathcal{P}$ particles, which are sampled from a proposal distribution $q(b_{1:N}|t_{1:N}, u_{1:N}, \boldsymbol{\omega}_{1:N})$. To infer the global parameters, μ_u and α_ℓ, we follow the literature in SMC devoted to the estimation of a static parameter [7, 8]. Specifically, we sequentially update the former by using maximum likelihood estimation and the latter by sampling from its posterior distribution.

Time complexity. The inference algorithm, which is detailed in Appendix A, has complexity $O(\mathcal{P}(\mathcal{U}+L+K)+\mathcal{P})$ per observed learning action, where L and K are respectively the number of learning patterns and the number of tasks uncovered up to this action. The only difference between this algorithm and the inference algorithm for HDP lies in Equation 10 and specifically in the computation of $\lambda^*_{u_n,k}(t_n)$. However, as we mentioned above, this value can be efficiently computed by using the properties of the exponential kernel function. For an experimental comparison of the two, we refer the reader to Section 4.2.

4. EXPERIMENTS

4.1 Experiments on synthetic data

In this section, we experiment with synthetic data and show that our inference algorithm can accurately recover the model parameters as well as assign each generated learning action to the true learning pattern given only the times and content of the learning actions.



(a) Dynamics (b) Content

Figure 3: Goodness of fit of the HDHP model in terms of (a) dynamics and (b) content.

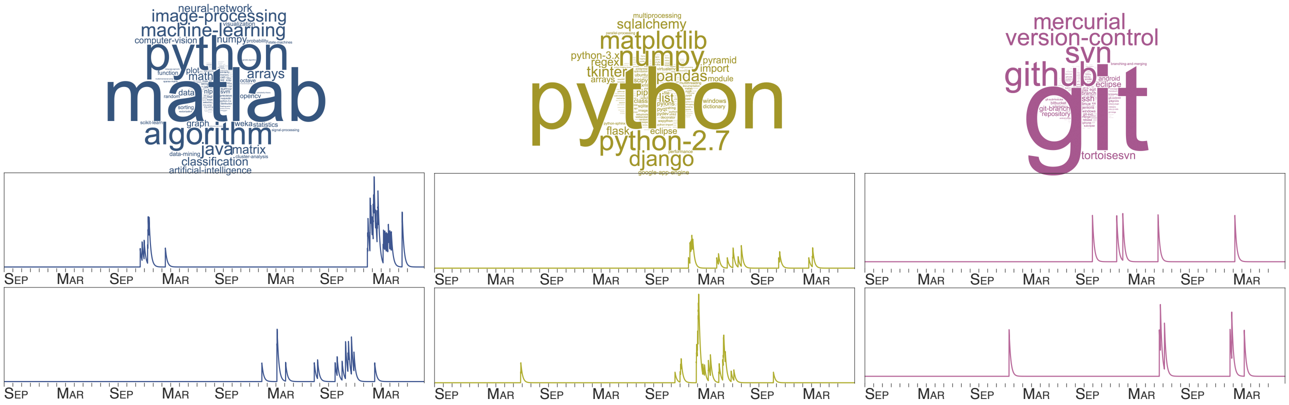
Experimental setup. We assume a set of 200 users, $L = 50$ learning patterns and a vocabulary of size $|\mathcal{W}| = 100$. We then sample the base intensity of each user μ_u from $Gamma(10, 0.2)$, and the learning pattern popularity vector $\boldsymbol{\pi}$ from a Dirichlet distribution with concentration parameters equal to 1. For each learning pattern, we sample the kernel parameter α_ℓ from $Gamma(8, 0.25)$, we randomly pick 30 words that will be used by the pattern and sample their distribution from a *Dirichlet* distribution with parameters equal to 3. We assume a kernel decay of $\nu = 5$. Then, for each user we generate online learning actions from the corresponding multivariate Hawkes process.

Results. Figures 2a and 2b summarize the results by showing the true and the estimated values of the base intensity of each user μ_u and the kernel parameter of each pattern α_ℓ respectively, using a total of 150k actions. In the latter, we match the inferred learning patterns to the true ones by picking the pair for which the NMI score is maximized. Moreover, Figure 2c shows the normalized mutual information (NMI) between the true and inferred clusters of actions against the number of actions seen by our inference algorithm. In all the above, we report the results for the particle which provided the maximum likelihood. It is clear that our inference algorithm accurately recovers the model parameters. As expected, using more actions when inferring the model parameters leads to more accurate assignment of actions to learning patterns.

4.2 Experiments on real data

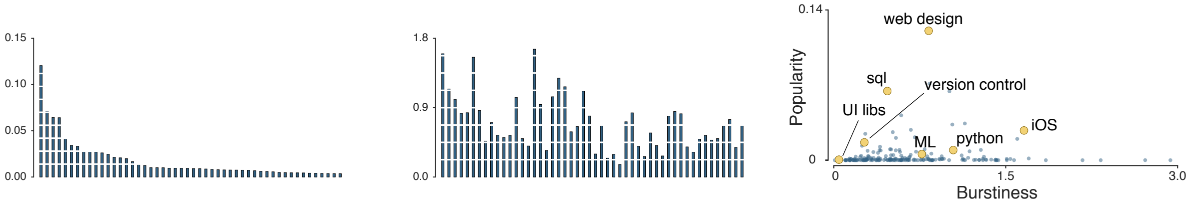
In this section, we experiment with real data gathered from Stack Overflow, a popular question answering (Q&A) site, where users can post questions—with topics ranging from C# programming to Bayesian nonparametrics—which are, in turn, answered by other users of the site. We apply our inference method on a large set of learning actions, and show that the inferred HDHP recovers meaningful learning patterns, accurately tracks users' interests over time, and provides better predictive performance than the HDP [19].

Experimental setup. We gather the times and content of all the questions posted by all Stack Overflow users during a four year period, from September 1, 2010 to September 1 2014. Here, we consider each user's question as a learning action. The reason for this choice is primarily the availability of public datasets, and, secondarily, the fact that a question provides clear evidence of the user's current interest at the time of asking. By looking only at the questions, we are underestimating the number of actions taken on each task, however, this bias is shared across all the tasks and, thus, we can still compare the dynamics of different patterns in a sensible way. For each question, we use the set of (up to) five tags (or keywords) that the user used to describe her



(a) Machine Learning (b) Python (c) Version Control

Figure 4: Three inferred learning patterns in Stack Overflow. The top row shows the content associated to each pattern, in the form of clouds of words, while the bottom row shows two samples of its characteristic temporal dynamics, by means of the intensities of two users using the pattern.



(a) Popularity (b) Burstiness (c) Popularity vs. Burstiness

Figure 5: Learning patterns. Panels (a) and (b) show the popularity and burstiness of the top-50 most popular learning patterns, and panel (c) shows the popularity and burstiness for all the inferred patterns. We highlight the learning pattern examples in Figure 4, as well as some others from Table 1.

question as the content associated to the learning action. To ensure that the inferred parameters are reliable and accurate, we only consider users who posted at least 50 questions and tags that were used in at least 100 questions. After these preprocessing steps, our dataset consists of ∼1.6 million questions performed by ∼16,000 users, and a vocabulary of ∼31,400 tags. Finally, we run our inference algorithm on the first 45 months of data and evaluate its performance on the last three months, used as held-out set.

Runtime. In our experiments, we use $|\mathcal{P}| = 200$ particles, and set $\beta = 1$ and $\nu = 5$ by cross-validation (based on the achieved log-likelihhod). Our implementation of the SMC algorithms for the proposed HDHP and the HDP requires, respectively, **71ms** and **65ms** per question on average, which implies that accounting for the temporal information in the data leads to an increase in runtime of only ∼10%.

Goodness of fit. In order to measure the performance of our proposed model, we evaluate its goodness of fit in terms of both content and temporal dynamics. To this end, we first evaluate the performance of the HDHP at capturing the temporal dynamics of the learning activity and compare it with the standard Hawkes process, which only accounts for the temporal information of the data and, therefore, cannot cluster learning actions into learning patterns. For the Hawkes process, we model the learning activity of each user as an independent univariate process, disregarding the content of each learning action. In other words, we learn both a base intensity μ and a self-excitation parameter α, as defined in Eq. 3, per user active in the test set $\mathcal{U}_{test}$. In order to compare the models' performance, we first apply the time changing theorem [9], which states that the integral of the intensity of a point process between two consecutive actions should follow a unit-rate exponential distribution. Then, we resort to two goodness of fit tests, the Kolmogorov-Smirnov and the Anderson-Darling [17], to measure how well the transformed action times fit the target distribution. Figure 3a summarizes the results by showing the percentage of the users in the held-out set that each test rejects at a significance level of 5%. From these results we can conclude that: i) a exponential time decay kernel provides a good fitting for the online learning activity for the vast majority of the users; and ii) while the Hawkes process performs slightly better (5% for the KS-test and 11% for the AD-test) than our model, it does so by using almost 2× more parameters ($2|\mathcal{U}_{test}| \sim 5$k for the Hawkes vs $|\mathcal{U}_{test}| + L^* \sim 2.7$k for the HDHP, where $L^* = 227$ is the number of inferred learning patterns). Moreover, we stress that since the Hawkes process, as opposed to the HDHP, does not allow us to cluster users' learning actions into learning patterns, it cannot be used to track users' interests over time, which is the main motivation of this work.

Second, we focus on evaluating the performance of the HDHP at clustering learning activity into learning patterns, and compare it with the HDP [19], which only makes use of the content information in the data. We resort to the marginal likelihood of the inferred parameters evaluated on the held-out set of questions $\boldsymbol{\omega}_i \in \mathcal{D}_{test}$, *i.e.*,

$$p(\boldsymbol{\omega}_i|\mathcal{D}_{train}, u_i, t_i) = \sum_{\ell=1}^{L} p(\boldsymbol{\omega}_i|\mathcal{D}_{train}, \ell) p(\ell|\mathcal{D}_{train}, u_i, t_i).$$

Above, the first term is defined in the same way for both models. However, for the HDP, the second term is simply the topic popularity π_ℓ, while for the HDHP it depends on the complete user history up to but not including t_i, *i.e.*, $p(\ell|\mathcal{D}_{train}, u_i, t_i) \propto \lambda^*_{u_i,\ell}(t_i)$, where $\lambda^*_{u_i,\ell}(t_i)$ is given by Eq. 6. Figure 3b shows the log-likelihood values obtained under the proposed HDHP and the HDP on the held-out set. Here, higher log-likelihood values mean better goodness of fit and, therefore, all the points above the $x = y$ line correspond to questions that are better captured by the HDHP, which are in turn 60% of the held-out questions. Additionally, we also compute the perplexity [6] on the held-out set as

$$perplexity = \exp\left\{-\frac{\sum_{i:e_i\in\mathcal{D}_{test}} \log p(\boldsymbol{\omega}_i|\mathcal{D}_{train})}{|\mathcal{D}_{test}|}\right\}.$$

The perplexity values for the HDHP and the HDP are 204 and 243, respectively, where here lower perplexity values mean better predictive performance. These results show that by modeling temporal information, in addition to content information, the HDHP fits better the content in the data than the HDP (20% gain in perplexity), and therefore, provides more meaningful learning patterns in terms of content, at a low extra cost in terms of runtime (∼10%). As a result, the proposed HDHP allows us to accurately track users' interest over time, and compare different learning patterns in terms of both content and temporal dynamics (refer to next sections).

Learning patterns. In this section, our goal is to understand the characteristic properties of the learning patterns that Stack Overflow users follow for problem solving. For ease of presentation, we focus on three particular examples of learning patterns, '*Machine Learning*', '*Python*' and '*Version Control*', among the 227 we uncovered. Figure 4 compares the above mentioned patterns in terms of content, by means of word clouds, and in terms of temporal dynamics, by means of the learning pattern intensities associated to two different users active on each of the patterns. Notice that in the intensity plots, the y-axis is in the same scale for all the learning patterns to allow for easier comparisons. Here, we observe that: i) the cloud of words associated to each inferred learning pattern corresponds to meaningful topics; and ii) despite the stochastic nature of the temporal dynamics, the user intensities of the same learning pattern tend to exhibit striking similarities in terms of burstiness and periods of inactivity. For example, we observe that *Machine Learning* and *Python* tasks exhibit much larger bursts of actions than *Version Control*. A plausible explanation is that version control tasks tend to be more specific and simple, *e.g.*, resolving a conflict while merging versions, and thus can be quickly solved by performing one or just a few questions. On the contrary, a user interested in machine learning or Python may face more complex tasks whose solution requires asking several questions in a relatively short period of time.

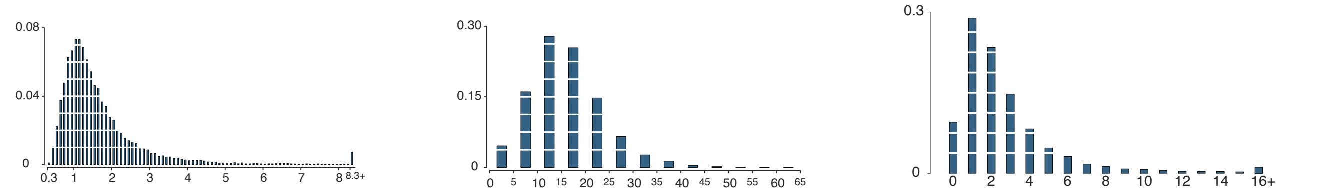
Top-20 most probable words in the learning pattern
'Web design': jquery javascript html php css ajax jquery-ui json forms arrays asp.net html5 jquery-mobile mysql dom regex jquery-plugins internet-explorer jquery-selectors wordpress
'sql': sql mysql sql-server php sql-server-2008 database tsql oracle postgresql sql-server-2005 database-design join stored-procedures c# select sqlite sql-server-2008-r2 java performance datetime
'iOS': ios objective-c iphone xcode cocoa-touch ipad uitableview cocoa core-data osx ios4 ios5 uiview uitableviewcell uiviewcontroller ios7 ios6 uinavigationcontroller uiscrollview nsstring
'Python': python numpy python-2.7 matplotlib django pandas python-3.x scipy tkinter flask sqlalchemy list arrays wxpython regex dictionary multithreading osx import google-app-engine
'Version control': git svn github version-control mercurial eclipse tortoisesvn merge branch repository ssh bitbucket xcode git-branch commit git-svn osx windows java gitignore
'Machine learning' (ML): matlab python algorithm r machine-learning java matrix plot artificial-intelligence numpy arrays image-processing nlp statistics opencv math octave data-mining scikit-learn neural-network
'UI Libraries': knockout.js javascript kendo-ui jquery asp.net-mvc knockout-2.0 kendo-grid durandal asp.net-mvc-4 knockout-mapping-plugin kendo-asp.net-mvc breeze single-page-application typescript mvvm asp.net-mvc-3 data-binding signalr json twitter-bootstrap

Table 1: The 20 most probable words for the seven patterns highlighted in Figure 5c.

Additionally, we investigate the relation between learning pattern popularity and burstiness, the latter being computed as the expected number of questions triggered by self-excitation during the first month after the adoption of the pattern. While popularity accounts for the probability of starting a new task on a learning pattern, the burstiness can be interpreted as a measure of how engaging a learning pattern is—more bursty patterns result in long sequences of closely related learning actions performed in short periods of time. Figure 5 shows the popularity and burstiness of the 50 most popular learning patterns sorted in decreasing order of popularity, as well as a scatter plot which shows the popularity against burstiness for all the inferred patterns. The figure reveals that the burstiness is not correlated with the popularity of a pattern. On the contrary, even among the top 20 most popular patterns, several of them trigger on average less than 0.5 follow-up questions. A possible explanation for this phenomenon is that the tasks associated with such patterns are more self-contained and the corresponding questions do not project towards a long-term goal. Figure 5c highlights examples of learning patterns that are very popular and bursty, *e.g.*, *Web design*; examples of bursty learning patterns that are not very popular, *e.g.*, *machine learning*; and learning patterns that are not popular nor bursty, *e.g.*, *UI libs*. Table 1 shows the top-20 most probable words in the seven learning patterns highlighted in Figure 5c.

Finally, it is also worth noticing that as shown by the popularity distribution in Figure 5a, which can be seen as the overall interests of the users of the online learning platform (*i.e.*, Stack Overflow), there is a small set of learning patterns which are much more popular than the rest. In particular, the most popular learning pattern, which is related to *Web design*, captures approximatively 12% of the attention of Stack Overflow users, and the 20 most popular learning patterns gather more than 60% of the popularity.

User behavior. In this section, we use our model to identify different types of users and derive insights about the learning patterns they use over time, as well as the evolution of their interests. Two natural questions emerge in this context: (i) do users stick to just a few learning patterns for all their tasks, or perhaps they explore a different pattern every time they start a task?; and, (ii) how long do they commit on their chosen task?

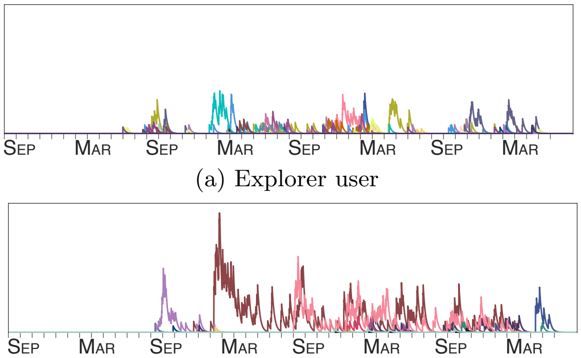


(a) Base intensity μ_u (tasks/month)

(b) # of adopted patterns per user

(c) Average time in months per task

Figure 6: User behavior: (a) the inferred user base intensities, (b) the number of learning patterns adopted by the users over the 4 years, and (c) the average time users spent for the completion of their tasks.



(a) Explorer user

(b) Loyal user

Figure 7: Real-world examples of user behavior. An explorer user (panel (a)), shifts over many different learning patterns over time, while a loyal user (panel (b)) sticks to a small selection of patterns.

First, we visualize the inferred intensities for two specific real users, among the several that we found, in Figures 7(a-b). These are examples of two very distinctive behaviors:

- ***Explorers***: They shift over many different learning patterns and rarely adopt the same pattern more than once. For example, the user in Figure 7a adopts over 10 patterns in less than a year, and rarely adopt the same learning pattern more than once.
- ***Loyals***: They remain loyal to a few learning patterns over the whole observation period. For example, the user in Figure 7b asks questions associated to two learning patterns over a period of 4 years period and rarely adopts new learning patterns.

We investigate to which extent we find explorers and loyals throughout Stack Overflow at large. To this end, we compute the user base intensities, μ_u, which can be viewed as the number of new tasks that a user starts per month, and the distribution of the total number of learning patterns adopted by each user over the observation period. Figures 6a and 6b summarize the results, showing several interesting patterns. First, there is a high variability across users in terms of new task rate—while most of users start one to two new tasks every month, there are users who start up to more than 8 tasks monthly. Second, while approximately 5% of the users remain loyal to at most 5 learning patterns and another 10% of the user explores more than 25 learning patterns over the 4 years, the average user (∼87%) adopts between 5 and 25 patterns during this period.

We also investigate how long users commit on their chosen task. To answer this question, we compute the average time between the initial and the final action for each task of our users. Figure 6c show the distribution of the average time spent per task. Here we observe that while approximately 10% of the user tasks are concluded in less than a month, most of the users (over 75% of the users) spend one to four months to complete a task.

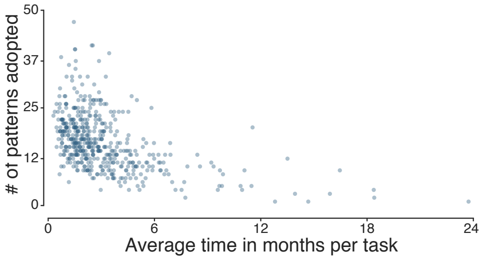


Figure 8: For a random sample of 500 users, we plot the number of different learning patterns that each user adopts vs the average time they invest in a task.

Finally, we explore in Figure 8 how long the users dedicate to a task compared to the number of unique learning patterns they adopt. Here, we observe that when users adopt many different learning patterns, they tend to spend only short time in each of their tasks. This behavior is hinting at a user being an *explorer*. On the other hand, more *loyal* users, i.e., the ones who adopt a limited number of different learning patterns, tend to spend more time in each of their tasks.

5. CONCLUSIONS

In this paper, we proposed a novel probabilistic model, the Hierarchical Dirichlet Hawkes Process (HDHP), for clustering grouped streaming data. In our application, each group corresponds to a specific user's learning activity. The clusters correspond to learning patterns, characterized by both the content and temporal information and shared across all users. We then developed an efficient inference algorithm, which scales linearly with the number of users and learning actions, and accurately recovers both the pattern associated with each learning user action and the model parameters. Our experiments on large-scale data from Stack Overflow show that the HDHP recovers meaningful learning patterns, both in terms of content and temporal dynamics, accurately tracks users' interests and goals over time, and provides better predictive performance than the state of the art.

Our work opens many interesting venues for future work. For example, the proposed HDHP could be run within the learning platform in an online fashion to track users' interest in real time and recommend questions that may be of interest at the right time. Although here we focused on modeling online learning activity data, it would be interesting to apply the proposed HDHP to cluster other types of streaming data, ranging from news articles, in which there is a single stream (group) of data, to web browsing, where one could identify groups of websites that provide similar services or content.

APPENDIX

A. DETAILS ON THE INFERENCE

Given a collection of n observed learning actions performed by the users of an online learning site during a time period $[0, T)$, our goal is to infer the learning patterns that these events belong to. To efficiently sample from the posterior distribution, we derive a sequential Monte Carlo (SMC) algorithm that exploits the temporal dependencies in the observed data to sequentially sample the latent variables associated with each learning action. In particular, the posterior distribution $p(b_{1:n}|t_{1:n}, u_{1:n}, q_{1:n})$ is sequentially approximated with a set of $|\mathcal{P}|$ particles, which are sampled from a proposal distribution that factorizes as

$$q(b_{1:n}|t_{1:n}, u_{1:n}, \boldsymbol{\omega}_{1:n})$$
$$= q(b_n|b_{1:n-1}, t_{1:n}, u_{1:n}, \boldsymbol{\omega}_{1:n})q(b_{1:n-1}|t_{1:n-1}, u_{1:n-1}, \boldsymbol{\omega}_{1:n-1}),$$

where

$$q(b_n|\boldsymbol{\omega}_{1:n}, t_{1:n}, u_{1:n})$$
$$= \frac{p(\boldsymbol{\omega}_n|\boldsymbol{\omega}_{1:n-1}, b_{1:n})p(b_n|t_{1:n}, u_{1:n}, b_{1:n-1})}{\sum_{(b_n)} p(\boldsymbol{\omega}_n|\boldsymbol{\omega}_{1:n-1}, b_{1:n})p(b_n|t_{1:n}, u_{1:n}, b_{1:n-1})} \qquad (12)$$

In the above expression, we can exploit the conjugacy between the multinomial and the Dirichlet distributions to integrate out the word distributions $\boldsymbol{\theta}_\ell$ and obtain the marginal likelihood $p(\boldsymbol{\omega}_n|\boldsymbol{\omega}_{1:n-1}, b_{1:n})$ For each particle p, the importance weight can be iteratively computed as

$$\mathrm{w}_n^{(p)} = \mathrm{w}_{n-1}^{(p)} p(t_n, u_n|t_{1:n-1}, u_{1:n-1}, b_{1:n-1}^{(p)}, \{\alpha_\ell^{(p)}\}_{\ell=1}^{L})Q_n^{(p)}, \qquad (13)$$

where $Q_n^{(p)} = \sum_{(b_n)} p(\boldsymbol{\omega}_n|\boldsymbol{\omega}_{1:n-1}, b_{1:n})p(b_n|t_{1:n}, u_{1:n}, b_{1:n-1}^{(p)})$ and $\mathrm{w}_1^{(p)} = 1/|\mathcal{P}|$. Since the likelihood term depends on the user base intensities μ_u and the kernel parameters $\{\alpha_\ell^{(p)}\}_{\ell=1}^{L}$, following the literature in SMC devoted to the estimation of a static parameter [7, 8], we infer these parameters in an online manner. In particular, we sample the kernel parameters from their posterior distribution up to, but not including, time t, and we update the user base intensities at time t as $\mu_u^{new} = r\mu_u^{old} + (1-r)\hat{\mu}_u$, where $\hat{\mu}_u$ is the maximum likelihood estimation of this parameter given the user history $\mathcal{H}_u(t)$ and $r \in [0, 1]$ is a factor that controls how much the updated parameter μ_u^{new} differs from its previous value μ_u^{old}.

Algorithm 1 summarizes the overall inference procedure, which presents complexity $O(\mathcal{P}(\mathcal{U}+L+K)+\mathcal{P})$ per learning action i fed to the algorithm, where L and K are the total number of learning patterns and tasks inferred up to the $(i-1)$-th action. Note also that, the for-loop across the particles $p \in \mathcal{P}$ can be parallelized, reducing the complexity per learning action to $O(\mathcal{U}+L+K+\mathcal{P})$.

Algorithm 1 Inference algorithm for the HDHP

```
Initialize $w_1^{(p)} = 1/|\mathcal{P}|$, $K^{(p)} = 0$ and $L^{(p)} = 0$ for all $p \in \mathcal{P}$.
for $i = 1, \ldots, n$ do
    for $p \in \mathcal{P}$ do
        Update the kernel parameters $\alpha_\ell^{(p)}$ for $\ell = 1, \ldots, L^{(p)}$ and the user base intensities $\mu_u^{(p)}$ for all $u \in \mathcal{U}$.
        Draw $b_i^{(p)}$ from (12).
        if $b_i^{(p)} = K^{(p)} + 1$ then
            Draw the new task parameters $\phi_{K^{(p)}+1}^{(p)}$ according to (11).
            Increase the number of tasks $K^{(p)} = K^{(p)} + 1$.
            if $\phi_{K^{(p)}+1}^{(p)} = \varphi_{L^{(p)}+1}^{(p)}$ then
                Draw the triggering kernel for the new topic $\alpha_{L^{(p)}+1}^{(p)}$ from the prior.
                Increase the total number of learning patterns $L^{(p)} = L^{(p)} + 1$.
        Update the particle weight $\mathrm{w}_i^{(p)}$ according to (13).
    Normalize particle weights.
    if $\|\mathbf{w}_i\|_2^2 < threshold$ then
        Resample particles.
```

6. REFERENCES

[1] O. Aalen, O. Borgan, and H. Gjessing. *Survival and event history analysis: a process point of view*. Springer Science & Business Media, 2008.

[2] A. Ahmed, Q. Ho, C. H. Teo, J. Eisenstein, A. J. Smola, and E. P. Xing. Online Inference for the Infinite Topic-Cluster Model: Storylines from Streaming Text. In *AISTATS*, 2011.

[3] A. Ahmed and E. Xing. Dynamic Non-Parametric Mixture Models and The Recurrent Chinese Restaurant Process with Applications to Evolutionary Clustering. In *SDM*, 2008.

[4] A.-L. Barabási. The origin of bursts and heavy tails in human dynamics. *Nature*, 2005.

[5] D. M. Blei and P. I. Frazier. Distance dependent chinese restaurant processes. *The Journal of Machine Learning Research*, 2011.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 2003.

[7] O. Cappé, S. J. Godsill, and E. Moulines. An overview of existing methods and recent advances in sequential Monte Carlo. *Proceedings of the IEEE*, 2007.

[8] C. Carvalho, M. S. Johannes, H. F. Lopes, and N. Polson. Particle learning and smoothing. *Statistical Science*, 2010.

[9] D. J. Daley and D. Vere-Jones. *An introduction to the theory of point processes. Vol. II*. Probability and its Applications (New York). Springer, second edition, 2008.

[10] N. Du, M. Farajtabar, A. Ahmed, A. J. Smola, and L. Song. Dirichlet-hawkes processes with applications to clustering continuous-time document streams. In *ACM SIGKDD*, 2015.

[11] M. Farajtabar, N. Du, M. Gomez-Rodriguez, I. Valera, L. Song, and H. Zha. Shaping social activity by incentivizing users. In *NIPS*, 2014.

[12] T. S. Ferguson. A bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1973.

[13] A. G. Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 1971.

[14] T. Iwata, A. Shah, and Z. Ghahramani. Discovering latent influence in online social activities via shared cascade poisson processes. In *ACM SIGKDD*, 2013.

[15] J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 1994.

[16] A. Smith, A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo methods in practice*. Springer Science & Business Media, 2013.

[17] M. A. Stephens. *Goodness of fit with special reference to tests for exponentiality*. Stanford University, 1978.

[18] Y. W. Teh and M. I. Jordan. Hierarchical Bayesian nonparametric models with applications. In *Bayesian Nonparametrics: Principles and Practice*. Cambridge University Press, 2010.

[19] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 2006.

[20] I. Valera and M. Gomez-Rodriguez. Modeling adoption and usage of competing products. In *ICDM*, 2015.

[21] Q. Zhao, M. A. Erdogdu, H. Y. He, A. Rajaraman, and J. Leskovec. Seismic: A self-exciting point process model for predicting tweet popularity. In *ACM SIGKDD*, 2015.

[22] K. Zhou, H. Zha, and L. Song. Learning triggering kernels for multi-dimensional hawkes processes. In *ICML*, 2013.