FM-Hawkes: A Hawkes Process Based Approach for Modeling Online Activity Correlations

Sha Li, Xiaofeng Gao*, Weiming Bao, Guihai Chen
Shanghai Jiao Tong University, Shanghai, China
Zoey.Lee@sjtu.edu.cn,gao-xf@cs.sjtu.edu.cn,wmbao24@gmail.com,gchen@cs.sjtu.edu.cn

ABSTRACT

Understanding and predicting user behavior on online platforms has proved to be of significant value, with applications spanning from targeted advertising, political campaigning, anomaly detection to user self-monitoring.

With the growing functionality and flexibility of online platforms, users can now accomplish a variety of tasks online. This advancement has rendered many previous works that focus on modeling a single type of activity obsolete.

In this work, we target this new problem by modeling the interplay between the time series of different types of activities and apply our model to predict future user behavior.

Our model, FM-Hawkes, stands for Fourier-based kernel multi-dimensional Hawkes process. Specifically, we model the multiple activity time series as a multi-dimensional Hawkes process. The correlations between different types of activities are then captured by the influence factor. As for the temporal triggering kernel, we observe that the intensity function consists of numerous kernel functions with time shift. Thus, we employ a Fourier transformation based non-parametric estimation. Our model is not bound to any particular platform and explicitly interprets the causal relationship between actions.

By applying our model to real-life datasets, we confirm that the mutual excitation effect between different activities prevails among users. Prediction results show our superiority over models that do not consider action types and flexible kernels.

KEYWORDS

User activity modeling, Time series analysis, Point processes

1 INTRODUCTION

Users often leave a trail of various activities on social network platforms or online marketplaces. For example, as illustrated in Figure 1, when we log on to our social networks accounts, we may choose to browse our timeline, click on some of the seemingly interesting links, reshare a portion of them, and perhaps follow a few new friends. When we go shopping on Amazon, we would typically search for the item, read through reviews, add some candidates to our wish list and after a final round of comparison, buy one of them. After the purchase happens, we might come back to leave a review.

Building accurate and interpretable temporal behavior models facilitates many applications, including targeted advertising on social platforms to maximize the probability of interaction [3], purchase prediction for marketing [14], user profiling [10], anomaly user detection [5, 6] and churn prediction [28]. Users may also record their daily activities and use behavior models for self-assessment.

Current studies on user behavior [5, 6, 14] focus on finding patterns in one type of activity, overlooking the fact that social platforms have grown to be much more powerful in terms of functionality. Actions may be supplementary, forming a workflow, such as committing new code and pushing it to the remote repository or substituting, indicating different options such as upvoting or downvoting an article on social rating website. Apparently, these actions are not independent, calling for a comprehensive model rather than studying them at an individual level.

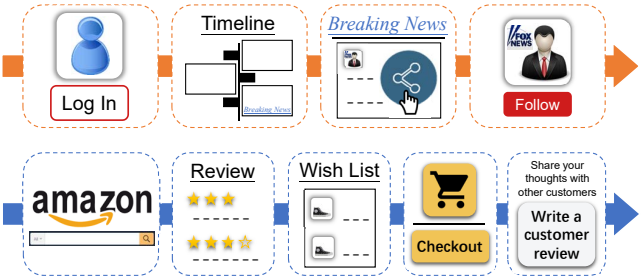


Figure 1: Actions forming a workflow

In this paper, we seek to build a unified framework for modeling multiple types of spontaneous behavior for a single user. Our method is not constrained to a particular platform but can learn relationships between multiple types of actions. By taking on a comprehensive perspective, we wish to uncover the interplay between different types of user behavior, predict future activity, and depict the patterns of regular user behavior.

Our task boils down to two core questions: 'if a user performs action A, will he go on to perform action B?' and 'when will the user perform action B?'. In essence, we are dealing with causal analysis and temporal prediction simultaneously. The former is related to the interpretability of the model, and the latter the accuracy and practical value.

Traditional time series analysis models such as ARIMA and their variations [23] deal with the latter problem by discretizing time through dividing the time series into equal length time intervals. They produce predictions in the form of "how many activities will

*Corresponding author.

CIKM'17 , November 6–10, 2017, Singapore, Singapore

ACM ISBN 978-1-4503-4918-5/17/11…$15.00
https://doi.org/10.1145/3132847.3132883

be performed in some future time interval". They also fail to reveal the causal correlation between succeeding events.

Point process models with Markovian state transfer [4, 5, 19] are capable of predicting the exact time of future activities on a continuous time scale. They consider the impact of the direct predecessor but not long term effects.

To tackle the above problem, we purpose the FM-Hawkes model, short for Fourier-based kernel multi-dimensional Hawkes model. Following the previous success of point process models in event modeling, we also build our model under the framework of stochastic point process theory. However, we do not assume states for the user, turning to Hawkes processes for explicitly capturing the triggering effect of previous events on later events. Time series of users are modeled as multi-dimensional Hawkes processes with each type of action corresponding to a dimension. The correlation between actions is then captured by the influence factor between dimensions.

Unlike many previous works that assume the triggering effect follows the patterns of exponential or power-law decay, we learn the temporal relationship from data, eliminating the need for prior knowledge. Observing that the intensity is the sum of weighted triggering functions with time shift, we devise a novel method based on Fourier transformation for learning the triggering function. Compared to ODE based methods, our proposed method generates continuous functions with lower complexity.

Finally, we put our model to test by using it to fit actions on an online forum Metafilter and the development platform Github. By analyzing the learned parameters, we find that there are distinct patterns in the distribution of parameters, dividing users into two categories. We also attempt the task of predicting user or repository behavior by using an alternative version of the thinning algorithm [9, 22]. Prediction results show that our model outperforms baselines that do not differentiate activity types and use fixed parameterized kernels.

Our main contributions are:

(1) We build a unified framework, FM-Hawkes, for characterizing and predicting multiple types of spontaneous user behavior. Our model is extensible for use in all platforms and can learn the correlation between any two types of actions.
(2) Modeling actions as multi-dimensional Hawkes processes, we design a non-parametric estimation method for the triggering function that does not require any domain knowledge.
(3) Our model is capable of predicting future user activity by simulation. We demonstrate this ability with experiments on real-life datasets. Our results show that considering inter-category triggering effects produces more accurate predictions.

We conduct a brief review of previous research on online user activity modeling and Hawkes processes in Section 2. After introducing the terminology and notation in Section 3, we present the FM-Hawkes model in Section 4. We derive the EM algorithm for inferring parameters and introduce our Fourier-based Triggering function estimation procedure. Experiments are described in Section 5. Last of all, we conclude our work and discuss possible improvements in Section 6.

2 RELATED WORK

2.1 User Activity Modeling

One of the earliest works on modeling human behavior patterns is the negative binomial distribution model (NBD) [7]. It has been used to model purchase behavior in marketing studies. NBD assumed that the probability of purchase was stationary over time, which is overly simplistic for the complex behavior in online platforms today.

With the rise of the Internet, more studies aimed at characterizing online user behavior. [19] proposed a cascading non-homogeneous Poisson process to model email correspondence. This model was based on two observations, namely bimodal and periodical patterns. The model is equivalent to a hidden Markov model with two states.

[28] formulated the task as a binary classification problem for dropout prediction. [5] studied behavioral patterns from the perspective of inter-activity time (IAT). They identified four major characteristics of IAT: positive correlation between successive IATs, periodic spikes, bimodal distribution and heavy-tailed distribution. Based on this discovery, Alceu et al. proposed the RSC model which stands for Rest-Sleep-Comment, the three user states. [4] reveals activity patterns in social voting websites by similar methodology. BuSca [6] model is a mixture of the Poisson process and the self-feeding process. The Poisson process component represents constant and regular behavior while the self-feeding component represents the burstiness. The Hierarchical Time-Rescaling Model (HTR) [14] also adopts the theory of point processes, and defines the rate function as the product of several fluctuation factors. By adjusting its parameters, Possionian, periodic, bursty and self-exciting patterns could be reproduced by HTR. However, the above works all limited their study to a single type of user activity.

[9] considered repost actions as triggers for following actions, thus leading to the co-evolution of social networks and information cascades. Their model is specifically designed for these two types of actions, and cannot be directly applied to other actions. The work most similar to ours, [23] proposed a general linear auto-regression model that considers activity smoothness, periodicity as well as interaction between different types of behavior. When dealing with sparse activities and short observed sequences, the model fails to produce accurate predictions.

2.2 Hawkes Processes

Hawkes processes [13] have been widely used to capture self-exciting and mutual-exciting behavior between entities. A few applications are predicting earthquakes, modeling financial markets and crime modeling [12, 16, 20].

In the field of social mining, Hawkes processes have been adopted to model information propagation on networks[24][2][26]. [21] combined a modified Hawkes process with feature engineering to predict the popularity of content. [17] also apply the marked Hawkes process to detect rumours.

Traditional choices for the triggering kernel include the exponential function and the power-law function. As for non-parametric kernel estimation, [27] and [18] proposed a non-parametric model based on ordinary differential equations.[1] used an estimation based on solving the Wiener-Hopf equation. Contrast function-based estimation in [11], also leaded to the least squares problem [8].

Both [15] and [25] decomposed the triggering kernel into a set of basis functions and estimated the coefficients.

3 PRELIMINARY

Definition 1. An *activity* (*action*) is defined as a triple $i = (u, t, c)$ where u is the user, t is the time of the action and c is the type or category of the action.

Lower letters i, j and l represent activities, and the corresponding user, time and activity category are denoted as u_i, t_i and c_i respectively. We use *activity* and *action* interchangeably throughout our paper.

Definition 2. An *activity sequence* is the set of activities performed by user u up to observation time $t (0 \le t \le T^u)$, $S^u(t) = \{(u, c_i, t_i)\}_{i=1}^{N^u(t)}$, as depicted in Figure 2. The length of this sequence $|S^u(t)| = N^u(t)$.

Every activity sequence consists of subsequences of specific types of activities. We regard to the subsequence that contains only type c activities as $S_c^u(t)$. Similarity, we have $|S_c^u(t)| = N_c^u(t)$.

When $t = T^u$, we drop the time t to simplify the notation. For example, $S^u(T^u) = S^u$ and $N_c^u(T^u) = n_c^u$. Note that the terminal time of observation T^u often exceeds the time of the last activity. Observations for all types of activities of one user are aligned, which means $T_c^u = T^u$ for all c.

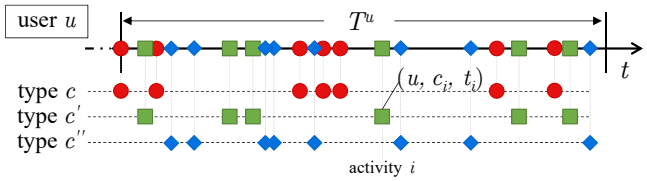


Figure 2: Illustration of activity sequence

Definition 3. A *temporal point process* is a list of discrete events occurring in continuous time $\{t_1, t_2, \cdots, t_n\}$ with $0 \le t_i \le T$.

Alternatively, the same information can be represented as a counting process, where we record the number of events that have occurred up to a time point t by random variable $N(t)$. The definition of temporal point processes has made it particularly amenable for modeling random events in time, with wide applications in queuing theory, computational neuroscience, telecommunication network modeling and human interaction modeling. In this paper, we represent multi-category user activities as a multi-dimensional point process.

Definition 4. The *conditional intensity function* (*rate function*) $\lambda(t)$ of a point process is the expected instantaneous rate of events occurring given the history.

$$\lambda_c^u(t)dt = \lambda_u^c(t|S^u(t))dt = E[dN_c^u(t)|\mathcal{S}^u(t)] \quad (1)$$

A point process is completely characterized by its conditional intensity function. However, given its stochastic nature, a point process may have many realizations. $\mathcal{S}^u(t)$ is the set of all possible realizations of $S^u(t)$.

Given a collection of action sequences $\{S^u\}$ from user group $\mathcal{U}$ with activity types C, we wish to accomplish the tasks listed below:

Task 1: Uncover the correlation between different types of user activity.

We wish to quantify to what extent does an action j of type c' affect the probability of user u performing an action i of type c and how this effect changes with time. Under the framework of point processes, we need to estimate $\Delta\lambda_c^u(t)$ with $S^u(t)$ and $S^u(t) - \{j\}$.

Task 2: Activity Prediction.

Given a user's action sequence S^u up to T^u, predict $N_c^u(t)$ for any time in the future.

We list the symbols used throughout the paper in Table 1.

Table 1: List of Symbols

Symbol	Definition
$N_c^u(t)$	the number of actions of type c that u has performed up to t
$S^u(t)$	action sequence of u up to t
n_c^u	the total number of actions u performed of type c
T^u	the total time user u has been observed
C	the set of actions an user can perform
$\mathcal{U}$	the set of users
$\lambda_c^u(t)$	conditional intensity function for u of type c
μ_c^u	the base intensity for action c of user u
$\phi_{c'c}^u$	the kernel function between actions c' and c for user u
$a_{c'c}^u$	the influence factor for action c' to action c for user u
$\kappa(t)$	the global triggering function
N	the number of time slots in $N(t)dt$
M	the total number of actions recorded

4 FM-HAWKES MODEL

In this section we describe our model FM-Hawkes in detail. We will first introduce Hawkes processes and its multi-dimensional extension. In FM-Hawkes, we tailor the parameters of multi-dimension Hawkes processes for the task of modeling multiple types of online user behavior. Based on this setting, we derive an EM algorithm to learn the user-specific parameters in FM-Hawkes. The key is to explicitly represent the branching process or causal relationship between activities by introducing latent variables. We are then left with the temporal kernel. Given the observation that the intensity function is actually composed by the temporal kernel shifted in time, we simplify the problem by transforming the triggering function into the frequency domain and perform Fourier based kernel estimation.

We also performed a thorough complexity analysis of the steps in our iterative algorithm. Compared to another state-of-the-art non-parametric kernel estimation method that is based on solving ODEs(Ordinary Differential Equations), our algorithm has much less complexity.

In the last subsection, we show how our method can generate predictions. Due to the stochastic nature of point processes, our prediction is completed in the form of sampling new activities and updating the intensity function. By performing multiple simulations, we can get a good estimation of the number of activities that a user would perform in a future period.

4.1 Multi-Dimensional Hawkes Processes

In its one-dimensional form, a Hawkes process defined as a point process with the intensity function in the form of (2).

$$\lambda(t) = \mu + \sum_i \phi(t - t_i) \tag{2}$$

The intensity consists of two parts, the base intensity μ and the excitation from previous events $\phi(t - t_i)$.

Due to the need to consider multiple types of activities, we extend this basic Hawkes process to a multi-dimension form, with one type of activity corresponding to a type of its own. Thus the intensity function is defined as (3).

$$\lambda_c^u(t) = \mu_c^u(t) + \sum_{t_i < t} \phi_{c_i c}^u(t - t_i) \tag{3}$$

The function $\phi_{c'c}^u(t)$ represents the influence for user u from an activity of type c' to another activity of type c. The strength of the influence changes over time, in some form of temporal decay.

It is noteworthy that the impact between two action types may not be symmetric. For example, actions that are performed in a fixed sequence, such as searching on an online marketplace and adding an item to the shopping cart, would not be performed in the reversed order. So the influence factor for search → purchase would not be equal to the influence factor for purchase → search.

In our model, we decompose the kernel function $\phi_{cc'}^u(t)$ into $a_{cc'}^u \kappa(t)$.

$$\lambda_c^u(t) = \mu_c^u + \sum_{t' < t} a_{cc'}^u \kappa(t - t') \tag{4}$$

By doing so, we separate the user and type variables with the time variable. This is critical to the inference of the model parameters, but also can be justified in terms of meaning.

We believe that users have different usage habits: the normal, periodical behavior is captured by base intensities μ_c^u and bursty behavior is described by correlation coefficients $a_{c'c}^u$.

Since the functionality of the platform is the same for all users, the temporal decay effect should be similar. Instant message platform should display more rapid decay compared to work-related platforms.

Summing up, the FM-Hawkes model has the following parameters:

- A global triggering function $\kappa(t)$ that describes the temporal decay of the excitation effect between activities.
- Influence factors that describe inter-category correlation $\{a_{cc'}^u\}$ and base intensities that describe regular behavior $\{\mu_c^u\}$. These parameters are local to the user.

4.2 EM Algorithm

The Hawkes model can be seen as a mixture model that generates actions either from the excitation of previous actions, or the base intensity.

The generation process for event i of type c_i at time t_i is as follows:

(1) Draw from all $N^u(t_i)$ previous user events and the base intensity the cause of event i with equal probability $\frac{1}{N^u(t_i)+1}$.
(2) If event i is caused by another event j, then event i is generated with intensity $a_{c_j c_i}^u \kappa(t_i - t_j)$.
(3) Otherwise the event i is intrinsic to the user and caused by the base intensity $\mu_{c_i}^u$.

We therefore use latent variables p_{ij} and p_{ii} to represent the branching structure, in other words the causal relationship of the activities.

$$p_{ij} = \begin{cases} 1 \text{ if user u's action i was caused by event j} \\ 0 \text{ otherwise} \end{cases} \tag{5}$$

$$p_{ii} = \begin{cases} 1 \text{ if user u's action i is caused by base intensity} \\ 0 \text{ otherwise} \end{cases} \tag{6}$$

It is then straightforward to estimate the parameters with respect to latent variables p_{ij} and p_{ii} with the EM (Expectation Maximization) algorithm.

Generally, for a mixture model of $k = \{1, 2, \cdots, m\}$ components, let $\theta = \{\theta_1, \theta_2, \cdots, \theta_m\}$ be the parameters related to each component. π_k represents the probability of selecting the kth component. z_i is the latent variable for instance i, indicating the component that actually generated the instance. x_i is the observed variable for instance i. To perform an estimation of the parameters, we wish to maximize the Equation (7).

$$\begin{aligned} Q(\theta, \theta^{t-1}) &= E[\sum_i \log Pr(x_i, z_i|\theta)] \\ &= \sum_i \sum_k Pr(z_i = k|\theta, x_i)(\log \pi_k + \log Pr(x_i|\theta_k)) \end{aligned} \tag{7}$$

E(Expectation) step:

According to the intensity function defined in Equation (3) and the Bayes theorem, for two actions i and j performed by the same user u, we have:

$$p_{ij} = \frac{a_{c_j c_i}^u \kappa(t_i - t_j)}{\lambda_c^u(t_i)} \tag{8}$$

$$p_{ii} = \frac{\mu_{c_i}^u}{\lambda_c^u(t_i)} \tag{9}$$

where $\lambda_c^u(t_i) = \mu_{c_i}^u + \sum_{t_l < t_i} a_{c_l c_i}^u \kappa(t_i - t_l)$.

M(Maximization) step: Once we have estimates for the latent variables, we can now try to find the parameters θ that will maximize the complete likelihood of data.

$$\theta = \text{argmax}_\theta Q(\theta, \theta^{t-1}) \tag{10}$$

In our case, since $\pi_k = \frac{1}{N^u(t)+1}$ is fixed given the data, maximizing Q is equivalent to $\max_\theta \sum_i \sum_k Pr(z_i = k|\theta, x_i) \log Pr(x_i|\theta_k)$.

$$
\begin{aligned}
&\max_{\theta} \sum_i \sum_k Pr(z_i = k|\theta, x_i) \log Pr(x_i|\theta_k) \\
&= \sum_u \sum_i^{n_u} \sum_k Pr(z_i = k|\theta, x_i) \log Pr(x_i|\theta_k) \\
&= \sum_u \sum_i^{n_u} \sum_{t_j < t_i} p_{ij} \{\log Pr(x_i|a^u_{c_j c_i}, \kappa(t))\} \\
&+ \sum_u \sum_i^{n^u} p_{ii} \{\log Pr(x_i|\mu^u_{c_i})\} \\
&= \sum_u \sum_i^{n_u} [\sum_{t_j < t_i} p_{ij} \log a^u_{c_j c_i} \kappa(t_i - t_j) \\
&- \sum_c a^u_{c'c} \int_{t^u_i}^{T^u} \kappa(t - t^u_i) dt] \\
&+ \sum_u [\sum_i^{n_u} p_{ii} \log \mu^u_{c_i} - \sum_c \mu^u_c T^u] \qquad (11)
\end{aligned}
$$

The update values for μ^u_c and $a^u_{c\prime c}$ are then computed by setting the partial derivative to zero.

$$
\frac{\partial Q}{\partial \mu^u_c} = \sum_i^{n^u_c} p_{ii} (\frac{1}{\mu^u_c}) - T^u = 0 \qquad (12)
$$

$$
\mu^u_c = \frac{1}{T^u} \sum_i^{n^u_c} p_{ii} \qquad (13)
$$

$$
\frac{\partial Q}{\partial a^u_{c'c}} = \sum_i^{n^u_c} \sum_{t_j < t_i \wedge c_j = c'} p_{ij} \frac{1}{a^u_{c'c}} - \sum_i^{n^u_c} \int_{t_i}^{T^u} \kappa(t - t_i) dt = 0 \qquad (14)
$$

$$
a^u_{c'c} = \frac{\sum_i^{n^u_c} \sum_{t_j < t_i \wedge c_j = c'} p_{ij}}{\sum_i^{n^u_c} \int_{t_i}^{T^u} \kappa(t - t_i) dt} \qquad (15)
$$

It is noteworthy that the estimation of the base intensity μ^u_c is not dependent on the triggering function $\kappa(t)$, but updating the influence factors $a^u_{c\prime c}$ requires computing the integral of $\kappa(t)$.

In the initialization stage, we first set all p_{ii} and p_{ij} to be equal for every activity i. For the first round of computing $a^u_{c\prime c}$, since $\kappa(t)$ is still unknown at the moment, we set the denominator to n^u_c.

4.3 Fourier Based Triggering Function Estimation

As seen in the section above, we are left to estimate the triggering function $\kappa(t)$ that characterizes how the intensity of self and mutual excitation decays over time. Unlike previous works that assume the kernel takes on a particular form, we wish to estimate $\kappa(t)$ in a non-parameter fashion, so that no prior knowledge is required and our model can adapt to various types of social platforms.

However, when the triggering coefficients $\{a^u_{c'c}\}$ are unknown, we cannot directly observe $\kappa(t)$. However, we can obtain samples of $\lambda(t)$, which is the weighted sum of time-shifted triggering functions as shown in Figure 3. We assume that all realizations of $\mathcal{S}^u$ exist

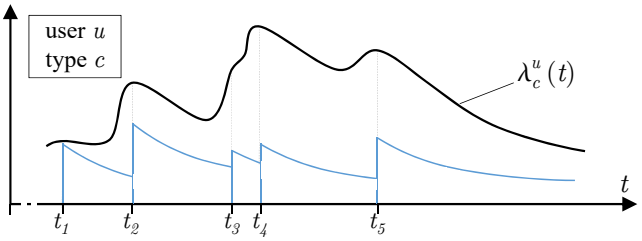


Figure 3: $\lambda(t)$ is consisted of $\kappa(t)$ with time shifts

in the network history. Thus we can use observations of activity counts as an approximation of the intensity function.

$$
\sum_{\mathcal{U}} \sum_C \lambda^u_c(t) dt \approx \sum_{\mathcal{U}} \sum_C dN^u_c(t) = N(t) dt \qquad (16)
$$

Our key insight is that by transforming $\lambda(t)$ to the frequency domain, the time shift can be converted into the multiplication of an exponential, greatly simplifying the problem. For a function $f(t)$ that is defined on the time domain and its frequency domain representation $F(w)$, we have the following Fourier transform pair:

$$
f(t - t') \leftrightarrow e^{-jwt'} F(w) \qquad (17)
$$

Given that $\kappa(t) = 0$ when $t \le 0$,

$$
\lambda^u_c(t) = \mu^u_c + \sum_{c'} a^u_{c'c} \sum_{c_l = c' \wedge t_l < t} \kappa(t - t_l) \qquad (18)
$$

The frequency domain counterpart of λ^u_c, denoted as Λ^u_c is:

$$
\begin{aligned}
\Lambda^u_c(w) &= \int_{-\infty}^{\infty} \lambda^u_c(t) e^{-jwt} dt \\
&= \mu^u_c \times 2\pi\delta(w) + \sum_{c'} a^u_{c'c} \sum_{c_l = c'} e^{-jwt_l} K(w)
\end{aligned} \qquad (19)
$$

As the events for one user in one activity category are sparse, we use the activities from the entire network to estimate $\kappa(t)$.

$$
\Lambda(w) = \sum_{\mathcal{U}} \sum_C \Lambda^u_c(w) \qquad (20)
$$

For practical consideration, we take the Discrete Fourier Transformation of the accumulated activity time series of the entire network. We take N samples from $N(t)dt$ at equal time intervals.

$$
w_k = \frac{2\pi}{N} k, k = 0, 1, \cdots, N \qquad (21)
$$

Let $i = \{0, 1, \cdots N-1\}$ be the indexes of the time slots. DFT gives us the frequency domain representation of $N(t)dt$, which is $\Lambda(w)$ as $\{(w_k, y[w_k])\}$.$y[w_k]$ contains information about the amplitude and phase of the sinusoid wave of frequency w_k.

$$
y[w_k] = \sum_{i=0}^{N-1} e^{-jw_k i} N(t) dt[i] \qquad (22)
$$

Figure 4 shows the result of DFT on the Metafilter dataset.

After performing Fourier transformation, for a given w_k we can estimate $K[w_k]$ as

$$
K[0] = \frac{y[0] - 2\pi \sum_{\mathcal{U}} \sum_C \mu^u_c}{\sum_{\mathcal{U}} \sum_C \sum_{c'} a^u_{c'c} n^u_{c'}} \qquad (23)
$$

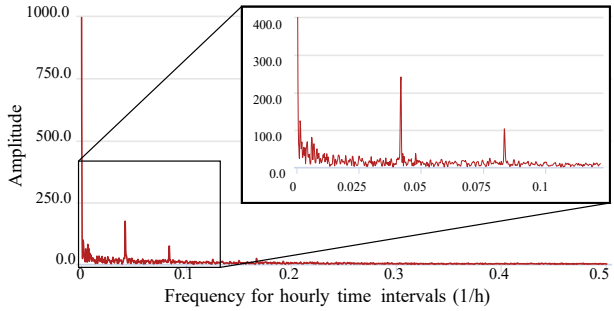


Figure 4: Fourier Transformation of $dN(t)$, one-sided spectrum

$$K[w_k] = \frac{y[w_k]}{\sum_{\mathcal{U}} \sum_{C} \sum_{c'} a^u_{c'c} \sum_{c_l=c'} e^{-jwt_l}} \tag{24}$$

Transforming back to the time domain yields

$$\kappa(t) = \frac{1}{N} \sum_{k=0}^{N-1} e^{jw_k t} K[w_k] \tag{25}$$

In our implementation, since the input 'signals' of DFT are real numbers, we only take the semi-positive frequencies. When transforming back to the time domain, we compute the complex conjugates of the signal values of the positive frequencies and pad them at the end. This ensures that the inverse DFT produces real values as well.

The complete algorithm for learn parameters in FM-Hawkes is listed below.

Algorithm 1: Learning parameters for FM-Hawkes

Input : $\{N(t)dt\}_{t=0}^{N}$
Output: $\{a^u_{c'c}\}$, $\{\mu^u_c\}$,$\kappa(t)$.

Initialize $\{p_{ij} = \frac{1}{N^u_c(t)+1}\}$, $\{p_{ii} = \frac{1}{N^u_c(t)+1}\}$.;
Initialize $\{\mu^u_c = \frac{1}{T^u} \sum_i^{n^u_c} p_{ii}\}$;
Initialize $\{a^u_{c'c} = \frac{\sum_i^{n^u_c} \sum_{t_j<t_i \wedge c_j=c'} p_{ij}}{n^u_c}\}$;
$\{(w_k, y[w_k])\} \leftarrow \text{DFT}(\{N(t)dt\})$. ;
while *not converge* **do**
  for *frequency* w_k **do**
    Calculate $K[w_k]$.
  Update p_{ii} and p_{ij} by Equation (9) and (8);
  Update $a^u_{c'c}$ and μ^u_c by Equation (15) and (13);

4.4 Complexity

In this section we take a closer look at the steps in our algorithm.

Discrete Fourier Transform is only performed once and is known to have the complexity of $N \log(N)$. Computation of $K[w_k]$ consists of $O(\sum_{\mathcal{U}}(|C|-1) \times (n^u))$ operations according to Equation (24). Since the number of types is a constant determined by the specific online platform, the complexity of $K[w_k]$ is linear to the total number of actions in the dataset $O(M)$. It is easy to see that $K[0]$ is of the same complexity. So updating $\kappa(t)$ needs $O(M \times N)$ operations.

In the E step, we need to update p_{ij} for every pair of activities and p_{ii} for every activity. As for the M step, μ^u_c requires a summation over p_{ii} and $a^u_{c'c}$ a selected summation over p_{ij}. The integral is actually computed through the following equation:

$$\int_{t_i}^{T^u} \kappa(t - t_i)dt = \frac{1}{N} \sum_{k=0}^{N-1} K[w_k] \frac{j}{w_k} (1 - e^{jw_k(T^u - t_i)}) \tag{26}$$

We list the computational complexity of major steps in our algorithm in Table 2.

Table 2: Complexity Analysis

Operation	Complexity for user	Complexity for network
DFT	-	$O(N \log N)$
$K[w_k]$	-	$O(M)$
Update $\kappa(t)$	-	$O(M \times N)$
Update p_{ij}	$O((n^u)^2 \times N)$	$O(\max\{n^u\} \times M \times N)$
Update p_{ii}	$O(n^u \times N)$	$O(M \times N)$
μ^u_c	$O(n^u)$	$O(M)$
$a^u_{c'c}$	$O((n^u)^2 + n^u \times N)$	$O(\max\{n^u\} \times M + M \times N)$

If the algorithm converges after L iterations, then the complete complexity is $O(L \times \max\{n^u\} \times M \times N)$.

It is easy to see that updating the latent variables p_{ij}, p_{ii} and parameters μ^u_c $a^u_{c'c}$ is embarrassingly parallel with respect to users. The triggering function $\kappa(t)$ is then updated by collecting the estimates of parameters for all users.

The ODE-based algorithm in [27] represents impact functions by N basis functions, where each basis function is discretized to K points. It learns basis functions and coefficients via alternating optimization. The complexity of the ODE-based algorithm per iteration is reported to be $O(Nn^{u3}U^2 + NK(n^uU + n^{u2}))$. It is noteworthy that in our algorithm, we represent the triggering function as a array of values on different frequencies rather than the exact function value at discrete time points.

4.5 Prediction

We predict future activities by simulation with a variation of the thinning process proposed in [9]. The key idea is to consider every dimension separately and draw a sample from each type of activity. Then the minimum time of the samples is a valid sample for the multi-dimensional Hawkes process. For one particular type of activity, rejection sampling is used to generate a sample since it is difficult to directly generate the waiting time from a Hawkes process. When no new activities are created, the Hawkes process degenerates to a non-homogeneous Poisson process. We assume that when no action is performed in $\{t, t + \Delta t\}$, the intensity $\lambda^u_c(t + \Delta t) \leq \lambda^u_c(t)$. Hence we can first draw a waiting time according to the exponential distribution and then test if the sample is valid.The procedure is described in detail in Algorithm 2.

Algorithm 2: Sampling the next activity

Input : Current time t, user u
Output: Time and type of next activity (s, c).

for *all types of activities* c *in* C **do**
  while *sample was rejected* **do**
    Calculate the intensity rate at t: $\lambda_c^u(t)$.;
    Draw waiting time according to Poisson process
    $p \sim Exp(\lambda_c^u(t))$.;
    Calculate the intensity rate at $t + p$:$\lambda_c^u(t + p)$.;
    Draw $q \sim Uniform(0, 1)$.;
    if *sample is rejected* $q\lambda_c^u(t) \geq \lambda_c^u(t + p)$ **then**
      $t \leftarrow t + p$.
    else
      $s_c \leftarrow t + p$.
$s \leftarrow \min_c s_c$;
$c \leftarrow argmin_c s_c$

5 EXPERIMENTS

In the following section, we will describe our dataset and parameter settings, then discuss empirical results and implications. Our model is implemented with MATLAB R2016a.

5.1 Datasets

5.1.1 Metafilter. Metafilter is a online forum that anyone can contribute a link or a comment to. Our Metafilter dataset was retrieved from Metafilter Infodump[1]. We focus on four activity types, *comments*, *posts*, *favorite* and *contact* as shown in Figure 5.

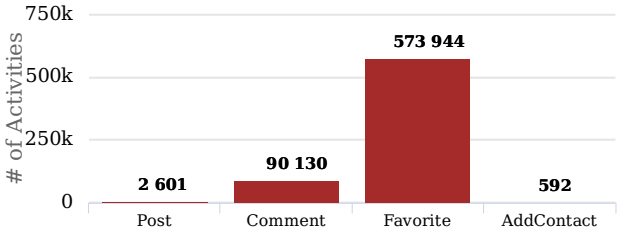


Figure 5: Frequency distribution on activity types (Metafilter)

5.1.2 Github. GitHub is a web-based Git repository hosting service with almost 20 million users and 57 million repositories up to April 2017.[2] Our data is drawn from GitHub Archive[3], which provides more than 20 activity types in the format of hourly aggregated archives. We select ten activity types and illustrate their frequency distribution in Figure 6. The exact size of our dataset is shown in Table 3.

[1]http://stuff.metafilter.com/infodump/
[2]https://github.com/blog/2345-celebrating-nine-years-of-github-with-an-anniversary-sale
[3]https://www.githubarchive.org/

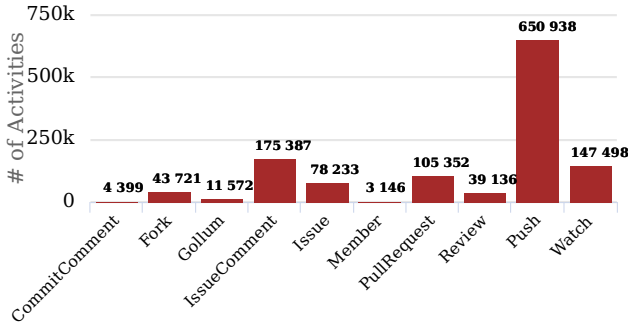


Figure 6: Frequency distribution on activity types (Github)

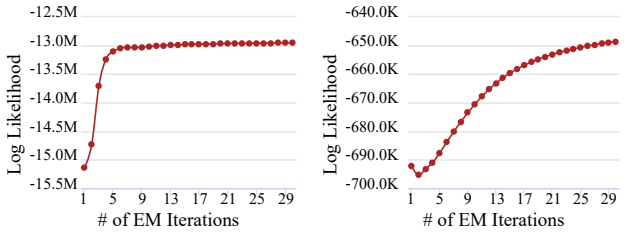
Table 3: Dataset Information

Name	# of users	# of activities	Month(s)	Year
Metafilter	5097	667,267	Nov. – Dec.	2016
GitHub	8375	1,259,382	Nov. – Dec.	2016

5.2 Parameter Setting

For discrete Fourier transform, we need to specify the sampling frequency f which is related to the number of time slots N in the estimation of $\lambda(t)$. The Nyquist rate for sampling states that $N > 2B$ where B is the bandwidth $\frac{w_{max}}{2\pi}$. We sample from $dN(t)$ of the entire network at one hour intervals.

Another consideration is the number of iteration we use in learning parameters and the number of times we run the simulation for prediction.



(a) Metafilter **(b) Github**

Figure 7: Convergence of EM Algorithm

Figure 7 shows the convergence rate on two datasets. The number of iterations required for convergence is quite related to the size of the data. In our experiments, we run the learning algorithm for 30 iterations.

Unlike the learning process, the predict is fairly stable over simulations. Hence, we perform 10 simulations for each period of prediction and take the average.

5.3 Pattern Discovery

We train our model on the entire dataset and look into the learned parameters and their implications.

The learned triggering function $\kappa(t)$ is displayed in Figure 8 and Figure 9.From the frequency domain graph, we can observe that

there are 4 conspicuous spikes, corresponding to 2.5 hour, 3 hour, 5 hour and 15 hour cycles. These may represent the intervals where users come back to the forum after their previous activity to check on updates. Converted to the time domain, the triggering kernel as shown in Figure 10 decays with time on the large scale in a smooth way, decrease much slower than exponential kernels assume.

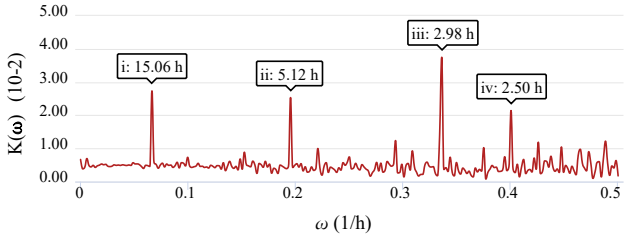


Figure 8: Triggering Function for Metafilter dataset

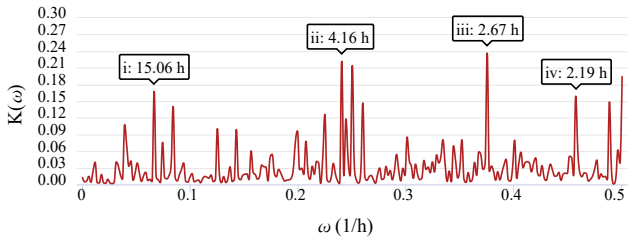
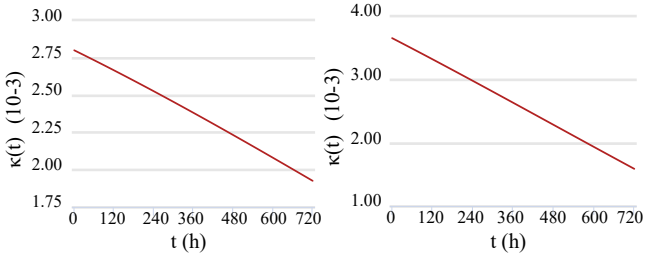


Figure 9: Triggering Function for GitHub dataset



(a) Metafilter **(b) Github**

Figure 10: Triggering Function $\kappa(t)$

While the triggering kernel is globally shared for all users in the online platform, the baseline intensity μ_c^u and $a_{c'c}^u$ are learned individually for different users. We first perform clustering by superKmeans in the Weka[4] toolbox using the learned user specific parameters as input features.

We discover that the users are distinctively divided into 2 clusters for both datasets. To interpret the cluster results, we select the top 4 features that are significant in the clustering procedure and plot the users' distribution in Figure 11 and 12.

On the Metafilter dataset, the major difference is whether posting new content would trigger succeeding events. The blue cluster, which is active in posting and commenting, also tends to mark

[4]http://www.cs.waikato.ac.nz/ml/weka/

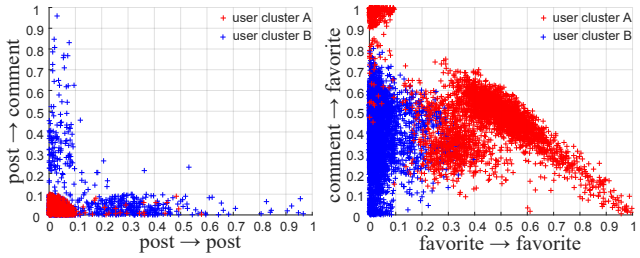


Figure 11: Clustering results for Metafilter dataset

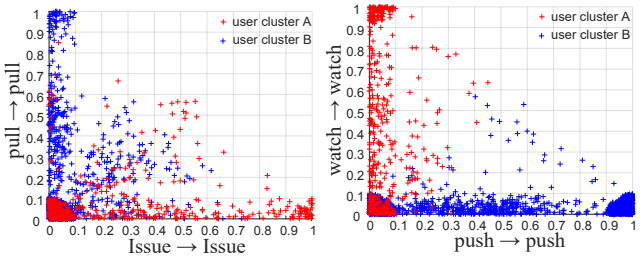


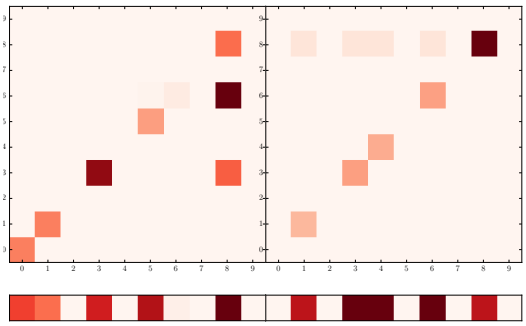
Figure 12: Clustering results for GitHub dataset

posts as favorite after participating in the discussion. However, their favorite actions always never results in a series of favorites. The red cluster, on the other hand, seldom posts at all, and users lie around the center of the figure, indicating relatively high comment → favorite and favorite → triggering effects. These users mostly play the role of "reader" and skim through the newly posted content, marking the ones that cater to their taste as favorite.

As for the clusters on Github, the red cluster is the more collaborative and popular type, with lots of discussions about current issues and many users attentively watching the development of the repo. The blue cluster is more personal and low-profile, focusing on uploading new code and keeping every member synchronized.

From each of the clusters, we select representative users and take a detailed look at the correlation between activities. The elements on the diagonal represent the self-triggering effects. In both clusters, the self-excitation plays a major role. Apart from the diagonal, the excitation matrix is quite sparse. On the left of Figure 13 is a "issue-driven" repo that has frequent discussions on issues (IssueComment) and these conversations lead to new pushes. Of course, there is apparently the need to pull from the repo before pushing new code. To the right of Figure 13 is an "commit-driven" repo that focuses on contributing new code to the repo on a frequent basis.

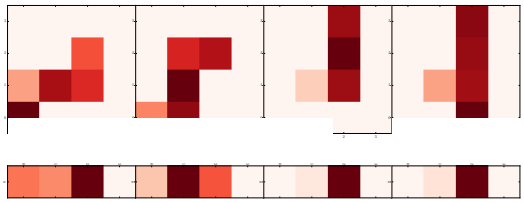
In Metafilter, we also see two types of users, namely "active" users and "passive" users. As shown in Figure 14, user 1 and user 2 are typical contributors of posts and comments. For user 2, commenting often leads him to further write a post or mark related posts as favorite.On the contrary, user 1's previous posting and faving behavior causes him to leave comments echoing his own opinion. Both of them do not add other users as contacts. For the "passive" users (user 3 and 4), they seldom create posts of their own, but often mark others' post as favorite, then join in the discussion



(a) Clus.A Repo (b) Clus.B Repo

Figure 13: Learned $a^u_{c'c}$ and μ^u_c (GitHub)

and meet new friends. The “add contact" functionality here is a loose analogy to Twitter’s follow, as users of Metafilter can use this action



(a) User 1 Clus.A (b) User 2 Clus.A (c) User 3 Clus. B (d) User 4 Clus. B

Figure 14: Learned $a^u_{c'c}$ and μ^u_c (Metafilter)

5.4 Activity Prediction

For the task of prediction, we divide the dataset into a training set of 50 days and hold out a test set of the remaining 11 days. Since FM-Hawkes is generative in nature, we can extend the prediction horizon by treating the generated sequence as observed history.

Our model generates temporal sequences for each user with activity type labels $\{(u, t, c)\}$. Other models may not exactly tell at which time point an action will occur, but assign the action to a time window. Therefore, to unify the evaluation of both types of models, we aggregate the prediction output to the day level.

Apart from FM-Hawkes, we implement the following models:

- **FM-Hawkes-reg.** We add a L1 norm regularization term for parameters $\{a^u_{c'c}\}$ to the loss function of FM-Hawkes. This is to reflect the belief that the correlations between actions should be sparse.
- **Exp-Hawkes.** We use an exponential function $\kappa(t) = we^{(-wt)}$ with a single parameter w as the triggering function. This imposes strong assumptions on the decay of influence between activities over time. The parameter w is also learned by the EM algorithm, updated every iteration along with μ^u_c and $a^u_{c'c}$.
- **FM-Hawkes-one.** This is an one-dimensional variation of our model. We treat all activities as the same type and train our model accordingly.
- **MAPer model.** This auto-regressive linear model was proposed in [23]. It explicitly considers the influence of previous actions, periodicity, and the influence between different types of activities.
- **MMHP model.** The multi-task multi-dimensional Hawkes process model estimates the triggering function at fixed time steps by solving the Euler-Lagrange equation [18] as initially proposed in [27]. The model contains 3 regularization terms to impose the smoothness of the triggering function and the low rank and sparsity of $\{a^u_{c'c}\}$.

We formally define the prediction task as to predict the number of actions performed by a user up to time t, namely $N^u_c(t)$. This is a numerical prediction task, and we evaluate our prediction accuracy by computing the mean square error (MSE) for prediction. The FuzzyMatch score is computed as the percentage of predictions that fall within a tolerance threshold of the actual action count. In the following experiments, we set the tolerance threshold to 1.0. We also attempt to classify whether users are active over the prediction period and therefore obtain accuracy, precision, recall and F1 scores for this task.

In the implementation of Exp-Hawkes, without putting restrictions on the shortest interval between activities, we found that the temporal clustering effect would often take over and generate an unrealistically large amount of activities.

Generally, FM-Hawkes and FM-Hawkes-reg performs well in terms of MSE and the F1 score. This shows that our model is good at simulating user behavior at a fine-grained level. FM-Hawkes-one performs equally well or even slightly better on classification. We reason that ’active’ is an attribute of the user and not of a particular action type, so modeling different actions types separately does not improve the performance. MAPer suffers from high MSE and low F1 score. The large number of linear coefficient parameters make the model very sensitive to records of rare action types and the error propagates when the prediction horizon is extended. On the other hand, when no activity is observed for a while, MAPer will continuously produce prediction of non-activity. MMHP has significantly high FuzzyMatch scores and high precision but low recall. This indicates thaht MMHP is underestimating the users’ activeness.

6 CONCLUSIONS & FUTURE WORK

In this paper we design FM-Hawkes, a novel model that captures the mutual influence between multiple types of user action in online platforms. Preceding our work, few have mentioned the correlation between different type of behavior and none have been devoted to devising such a model.

The multi-activity triggering model, is based on multi-dimensional Hawkes process, with each type of action corresponding to its own dimension. In order to let our model adjust to different platforms, we also purpose a novel Fourier transformation based non-parametric estimation of the triggering kernel.

Extensive experiments on real-life datasets show that this correlation effect prevails in user behavior and is useful for predicting future user activities.

In this work we only consider the correlations between the actions of a user and neglect the influence between pairs of users.

Table 4: Prediction results on Metafilter dataset

	Evaluation by type		Evaluation total					
Method	MSE	FuzzyMatch	MSE	FuzzyMatch	Accuracy	Precision	Recall	F1
FM-Hawkes	6.126	0.852	31.814	0.532	0.896	0.581	0.780	0.333
FM-Hawkes-reg	5.547	0.880	24.490	0.543	0.906	0.620	0.752	0.340
FM-Hawkes-one	-	-	27.272	0.536	0.905	0.614	0.778	0.343
Exp-Hawkes	460.927	0.828	1849.670	0.376	0.718	0.305	0.879	0.227
MAPer	8.594	0.855	38.546	0.475	0.767	0.121	0.119	0.060
MMHP	8.187	0.938	48.838	0.784	0.869	0.714	0.022	0.021

Table 5: Prediction results on Github dataset

	Evaluation by type		Evaluation total					
Method	MSE	FuzzyMatch	MSE	FuzzyMatch	Accuracy	Precision	Recall	F1
FM-Hawkes	5.228	0.912	62.395	0.205	0.649	0.288	0.611	0.196
FM-Hawkes-reg	5.030	0.919	57.776	0.233	0.680	0.305	0.570	0.198
FM-Hawkes-one	-	-	58.383	0.216	0.671	0.301	0.594	0.200
Exp-Hawkes	319.128	0.905	3201.058	0.155	0.450	0.211	0.722	0.163
MAPer	1.920×10^7	0.959	1.920×10^8	0.402	0.724	0.183	0.142	0.080
MMHP	21.698	0.969	1766.317	0.823	0.810	0.406	0.065	0.056

Directly adding the influence effect to the intensity function will greatly affect the algorithm's efficiency, so adjustments must be made to incorporate this factor. One possible solution is to limit the influence within community structures.

7 ACKNOWLEDGEMENTS

This work has been supported in part by the China 973 Project (2014CB340303), the Program of International S&T Cooperation (2016YFE0100300), China NSF Projects (Nos. 61672353, 61472252), and the CCF-Tencent Open Research Fund.

At the time of this paper, all authors are members of the Shanghai Key Laboratory of Scalable Computing and Systems, Department of Computer Science and Engineering, Shanghai Jiao Tong University, China.

REFERENCES

[1] Emmanuel Bacry, Khalil Dayri, and Jean-François Muzy. 2012. Non-parametric kernel estimation for symmetric Hawkes processes. Application to high frequency financial data. *EPJ B* 85, 5 (2012), 1–12.
[2] Peng Bao. 2016. Modeling and Predicting Popularity Dynamics via an Influence-based Self-Excited Hawkes Process. In *CIKM*. ACM, 1897–1900.
[3] Ye Chen, Dmitry Pavlov, and John F. Canny. 2009. Large-scale behavioral targeting. In *KDD*. 209–218.
[4] Alceu Ferraz Costa, Agma Juci Machado Traina, Caetano Traina Jr., and Christos Faloutsos. 2016. Vote-and-Comment: Modeling the Coevolution of User Interactions in Social Voting Web Sites. In *ICDM*. 91–100.
[5] Alceu Ferraz Costa, Yuto Yamaguchi, Agma J. M. Traina, Caetano Traina, and Christos Faloutsos. 2015. RSC: Mining and Modeling Temporal Activity in Social Media. In *KDD*.
[6] Rodrigo Augusto da Silva Alves, Renato Martins AssunÃğão, and Pedro O. S. Vaz de Melo. 2016. Burstiness Scale: A Parsimonious Model for Characterizing Random Series of Events. In *KDD*.
[7] A.S.C. Ehrenberg. 1959. The pattern of consumer purchases. *Applied Statistics* 8(1) (1959), 26–41.
[8] Michael Eichler, Rainer Dahlhaus, and Johannes Dueck. 2016. Graphical modeling for multivariate hawkes processes with nonparametric link functions. *J. Time Ser. Anal.* (2016).
[9] Mehrdad Farajtabar, Yichen Wang, Manuel Gomez-Rodriguez, Shuang Li, Hongyuan Zha, and Le Song. 2015. COEVOLVE: A Joint Point Process Model for Information Diffusion and Network Co-evolution. In *NIPS*.
[10] Scott A. Golder, Dennis M. Wilkinson, and Bernardo A. Huberman. 2007. Rhythms of social interaction: messaging within a massive online network. *C&T* (2007), 41–66.
[11] Niels Richard Hansen, Patricia Reynaud-Bouret, Vincent Rivoirard, et al. 2015. Lasso and probabilistic inequalities for multivariate point processes. *Bernoulli* 21, 1 (2015), 83–143.
[12] Stephen Hardiman, Nicolas Bercot, and Jean-Philippe Bouchaud. 2013. Critical reflexivity in financial markets: a Hawkes process analysis. (2013).
[13] Alan G Hawkes. 1971. Spectra of some self-exciting and mutually exciting point processes. *Biometrika* (1971), 83–90.
[14] Hideaki Kim, Noriko Takaya, and Hiroshi Sawada. 2014. Tracking Temporal Dynamics of Purchase Decisions via Hierarchical Time-Rescaling Model. In *CIKM*. 1389–1398.
[15] Remi Lemonnier and Nicolas Vayatis. 2014. Nonparametric markovian learning of triggering kernels for mutually exciting and mutually inhibiting multivariate hawkes processes. In *ECML PKDD*. 161–176.
[16] Erik Lewis, George Mohler, P Jeffrey Brantingham, and Andrea L Bertozzi. 2012. Self-exciting point process models of civilian deaths in Iraq. *Security Journal* 25, 3 (2012), 244–264.
[17] Michal Lukasik, P. K. Srijith, Duy Vu, Kalina Bontcheva, Arkaitz Zubiaga, and Trevor Cohn. 2016. Hawkes Processes for Continuous Time Sequence Classification: an Application to Rumour Stance Classification in Twitter. In *ACL (2)*. The Association for Computer Linguistics.
[18] Dixin Luo, Hongteng Xu, Yi Zhen, Xia Ning, Hongyuan Zha, Xiaokang Yang, and Wenjun Zhang. 2015. Multi-task multi-dimensional hawkes processes for modeling event sequences. In *IJCAI*. 3685–3691.
[19] R. Dean Malmgren, Jake M. Hofman, Luis A. Nunes Amaral, and Duncan J. Watts. 2009. Characterizing individual communication patterns. In *KDD*.
[20] David Marsan and Olivier Lengline. 2008. Extending earthquakes' reach through cascading. *Science* 319, 5866 (2008), 1076–1079.
[21] Swapnil Mishra, Marian-Andrei Rizoiu, and Lexing Xie. 2016. Feature Driven and Point Process Approaches for Popularity Prediction. In *CIKM*. ACM, 1069–1078.
[22] Yosihiko Ogata. 1981. On Lewis' simulation method for point processes. *IEEE Transactions on Information Theory* 27, 1 (1981), 23–31.
[23] Sarah Masud Preum, John A. Stankovic, and Yanjun Qi. 2015. MAPer: A Multi-scale Adaptive Personalized Model for Temporal Human Behavior Prediction. In *CIKM*. 433–442.
[24] Marian-Andrei Rizoiu, Lexing Xie, Scott Sanner, Manuel Cebrián, Honglin Yu, and Pascal Van Hentenryck. 2017. Expecting to be HIP: Hawkes Intensity Processes for Social Media Popularity. In *WWW*. ACM, 735–744.
[25] Hongteng Xu, Mehrdad Farajtabar, and Hongyuan Zha. 2016. Learning Granger Causality for Hawkes Processes. In *ICML*, Vol. 48. 1717–1726.
[26] Qingyuan Zhao, Murat A. Erdogdu, Hera Y. He, Anand Rajaraman, and Jure Leskovec. 2015. SEISMIC: A Self-Exciting Point Process Model for Predicting Tweet Popularity. In *KDD*. ACM, 1513–1522.
[27] Ke Zhou, Hongyuan Zha, and Le Song. 2013. Learning Triggering Kernels for Multi-dimensional Hawkes Processes. In *ICML*.
[28] Yin Zhu, Erheng Zhong, Sinno Jialin Pan, Xiao Wang, Minzhe Zhou, and Qiang Yang. 2013. Predicting user activity level in social networks. In *CIKM*. 159–168.