Networked Point Process Models Under the Lens of Scrutiny

Guilherme Borges ✉, Flavio Figueiredo,
Renato M. Assunção, and Pedro O.S. Vaz-de-Melo

Universidade Federal de Minas Gerais
{guilherme.borges, flaviovdf, assuncao, olmo}@dcc.ufmg.br

Abstract. Recently, there has been much work on the use of Networked Point Processes (NPPs) to extract the latent network structure of timestamp data. Several models currently exist to capture implicit interactions in hospital visits, blog posts, e-mail messages, among others. The problem is that evaluating these solutions is not a trivial task. First, the methods have only been evaluated in a few datasets by a limited number of metrics. Second, and even worse, the evaluation metrics are often unsuitable for the typically sparse networks, which consequently lead to inconclusive results. To provide the community with a rigorous benchmark, in this paper we propose an empirical evaluation framework of NPP models in the task of network extraction. We reevaluate several models of the literature using our framework and compare the results to two null models designed for this task. In our discussion, we point out when some methods should be used depending on the expected efficacy, execution time, or dataset properties. Overall, we find that only three models show consistent significant results in real-world data.
Source code: http://github.com/guilhermeresende/NPPs/

Keywords: Temporal point processes · Benchmarking · Network inference.

1 Introduction

Modeling the occurrence of events through temporal point processes is a necessity in many different fields. A temporal point process $\mathcal{P}$ is a stochastic process whose realizations consist of random event times $t_1, t_2, \cdots$, where each t_i is the time of the occurrence of the $i-$th event. Different point processes can be interrelated in such a way that events occurring in one process influence the occurrence of events in other processes [12]. In this case, the relationship among them can be encoded as an influence graph (or matrix) of Networked Point Processes (NPPs), where the nodes are the different processes, and the edge weights encode the influences from one node to the others [1, 5]. Inferring this influence matrix via timestamp data only [41] is a challenging task that has been studied in a variety of settings, such as hospital visits of multiple patients [10], viewing records of TV programs [41], and the publication times of articles across different websites [1, 45].

Following the growing popularity trend of Machine Learning (ML), several models for inferring the influence matrix of NPPs were recently proposed [1, 2, 15, 25, 41, 45, 46]. One problem is that, similarly to what is being reported for other fields within ML [20, 22,

27, 30, 35–38], the rate of empirical advancement of NPP models is not being followed by a consistent increase in the level of empirical rigor. Models are being evaluated using only a handful of efficacy metrics at best [1, 41, 45, 45], and most of these metrics are inadequate to assess sparse weighted networks, which is usually the case in NPPs. In addition, most studies evaluated their approaches using only a few real-world datasets, sometimes even only one [1]. Hyper-parameter sensitivity analyses are not conducted and almost no effort is shown to tune hyperparameters for baselines. Finally, very little is done to account for model limitations. Previous studies rarely report confidence intervals or comparisons with null models. As usual, in the field of ML, negative results are omitted in favor of wins. Sanity checks, such as verifying if the model is robust to false positives (i.e., i.i.d. data), are also ignored.

Motivated by these issues, and inspired by recent general guidelines for empirical rigor in the field of ML [35], we propose an empirical evaluation framework for NPP models. Our framework assumes that NPP models receive k series of timestamps (processes) as input and outputs the influence matrix among these k processes. This task is especially challenging because the input data is usually bursty [3], with sparse connectivity among processes [45], and may contain all sorts of noise (e.g., spams or bot messages) [44]. Also, the latent network structure may serve several purposes and, because of that, should be assessed and evaluated accordingly. While one may need the network to identify only the most influential nodes, another might require the whole network structure. Moreover, while some applications demand a low false-positive error rate, others are more concerned with false-negative errors. Thus, properly evaluating NPP models is not a trivial task.

Our proposed framework has five complementary fronts of evaluation, which can be applied to assess any data-driven model. First, we perform an analysis of how the likelihood of the model to the data correlates with evaluation metrics (e.g., NDCG). Second, we present a methodology to estimate an empirical upper-bound estimate based on the number of hyper-parameter configurations tested. Third, we evaluate the asymptotic complexity of training such methods and, fourth, we evaluate the models on seven real-world datasets using three complementary metrics. Finally, we assess the robustness of the models to false-positives via i.i.d. simulated data. In short, the main contributions of this work are:

- A comprehensive empirical evaluation framework for NPP models;
- A thorough comparison of state of the art NPP models;
- Guidelines for selecting NPP models for the task of network inference.

The rest of this work is organized as follows. In Section 2, we describe the related work. Next, in Section 3, we present the problem formulation and our empirical evaluation framework. Section 4 describes the results and, finally, in Section 5 we discuss the conclusions.

2 Related work

In a recent position paper, Sculley *et al.* [35] made a sound statement about how empirical rigor is not keeping pace with advances in Machine Learning (ML). The good news is

that, like us, many others are taking a step back and reviewing the current state of the art of different areas of ML through rigorous and thorough empirical evaluations [20, 22, 27, 30, 36–38]. A striking and common conclusion of these evaluations is that hyper-parameter tuning and testing multiple datasets can make traditional and most recent methods to perform equally. This was verified for generative adversarial networks [27], decision making [37], language models [30], information retrieval [22, 42], and clinical prediction [20].

Inspired by these works, we conduct an extensive evaluation of NPP models for network inference. Most previous work on NPPs falls into three classes: those that explore Hawkes Processes [21, 28, 31, 33], those that explore information cascades [7, 16–18, 32, 39], and those that explore Wold processes [15]. We argue that NPP models have also reached a point where there is a need to take a step back and evaluate models using a rigorous empirical methodology. Often, previous efforts to extract latent networks from data rely on fixed hyper-parameters [1] and test their methods on only a handful of real datasets [9, 14, 29, 43, 45, 46]. Also, previous works have employed a variety of distinct metrics and tools for validation, such as different simulated datasets, different problem definitions, and different error and ranking measures. Moreover, metrics do not usually transfer from one work to the other. For instance, [15] discussed how the Kendall correlations employed by [41, 43, 46] is unsuitable for sparse matrices.

3 The Evaluation Framework

We argue that a proper framework to evaluate the networked point process models must take into account: (1) a comparison with the state of the art models using (2) different metrics over (3) several datasets with (4) the proper use of confidence intervals and null models to test significance hypotheses. Throughout this section, we present the problem definition and our methodology to perform such evaluations.

3.1 Problem Definition

Consider multiple point processes $\mathcal{P}_a, \mathcal{P}_b, \cdots, \mathcal{P}_K$ observed simultaneously, where each event is associated to a single process: $\mathcal{P}_a = \{0 \leq t_{a_1} < t_{a_2} < \ldots\}$. Let $\mathcal{P} = \{1, \cdots, K\}$ be the set of all processes, where $|\mathcal{P}| = K$, i.e., we have a total of K processes or nodes. For each $a \in \mathcal{P}$, the counting process $N_a(t) = \sum_{i=1}^{|\mathcal{P}_a|} \mathbb{1}_{t_{a_i} \leq t}$ is the total number of events of $\mathcal{P}_a$ until time t. Finally, $N = |\bigcup_{a=1}^{K} \mathcal{P}_a|$ is is the total number of events considering the superimposed union of all events from all point processes.

Let $\mathcal{H}_a(t)$ be the history of the a-th point process up to time t, called the filtration of the process. A point process can be completely characterized by its conditional intensity function [12] defined as: $\lambda_a(t|\mathcal{H}_a(t)) = \lim_{h \to 0} \frac{\mathbb{P}(N_a(t+h) - N_a(t) > 0 | \mathcal{H}_a(t))}{h}$. Assuming no simultaneous events, this function defines the instantaneous probability of one event occurring at each time t given the entire previous history up to t. In a multivariate setting, we can extend this definition by conditioning the intensity $\lambda_a(t|\mathcal{H}_\mathcal{P}(t))$ on the history of all events $\mathcal{H}_\mathcal{P}(t)$. For instance, if we have three timestamps from two processes $t_{a1} < t_{b1} < t_{a2} < t$: $\mathcal{H}_\mathcal{P}(t) = \{t_{a1}, t_{b1}, t_{a2}\}$; $\mathcal{H}_a(t) = \{t_{a1}, t_{a2}\}$; and, $\mathcal{H}_b(t) = \{t_{b1}\}$. In other words, $\lambda_a(t|\mathcal{H}_a(t))$ captures the evolution of process $\mathcal{P}_a$ by focusing solely

Method	Complexity (per iteration)	Hyper-parameter range
ExpKern	$O(NK^2)$	$\beta \in [0, 10]$; $C \in [0, 10^3]$; $pnlty \in [l_1, l_2, nuc, none]$;$solver \in [gd, agd]$; $\alpha \in [10^{-4}, 1]$;$tol \in [10^{-8}, 10^{-4}]$
HkEM	$\Omega(NK^2)$	$support \in [1, 200]$; $size \in [1, 400]$; $tol \in [10^{-8}, 10^{-4}]$
ADM4	$O(N^3K^2)$	$\beta \in [0, 10]$; $C \in [0, 10^3]$; $lasso \in [0, 1]$; $tol \in [10^{-8}, 10^{-4}]$
MLE-SGP	$O(MN^3K^2)$	$max - mean - f \in [1, 200]$;$\#f \in [1, 20]$; $\alpha \in [10^{-9}, 10^{-3}]$; $C \in [0, 10^3]$; $lasso \in [0, 1]$; $tol \in [10^{-8}, 10^{-4}]$
HC	$O(K^3)$	$H \in [1, 200]$; $C \in [0, 10^3]$; $solver \in [adam, ada, rmsp, adad]$; $pnlty \in [l_1, l_2, none]$; $\alpha \in [10^{-4}, 1]$; $tol \in [10^{-8}, 10^{-4}]$
GB	$O(N(\log(N) + \log(K)))$	$\beta \in [1, 10]$
NetInf	—	$\alpha \in [0, 1]$; $m \in [exp, powerlaw, rayleigh]$

Table 1. Complexity and hyper-parameter ranges used for optimizing the models.

on $\mathcal{P}_a$'s history, $\lambda_a(t|\mathcal{H}_\mathcal{P}(t))$ states that a's evolution also depends on $\mathcal{P}_b$. This is our starting point for defining the *network structure* as follows.

Network Structure: Let $\mathcal{P}_a$ and $\mathcal{P}_b$ be two arbitrary processes in $\mathcal{P}$. Also, let $\mathcal{Q} = \mathcal{P} - \{\mathcal{P}_b\}$. When $\lambda_a(t|\mathcal{H}_\mathcal{Q}(t)) = \lambda_a(t|\mathcal{H}_\mathcal{P}(t))$ we can state that process $\mathcal{P}_b$ does not influence process $\mathcal{P}_a$. That is, the intensity function of a with or without $\mathcal{P}_b$ is, at least statistically, equivalent. When $\lambda_a(t|\mathcal{H}_\mathcal{Q}(t)) \neq \lambda_a(t|\mathcal{H}_\mathcal{P}(t))$, $\mathcal{P}_b$ influences $\mathcal{P}_a$, as $\mathcal{P}_b$'s past has some influence on the intensity function of $\mathcal{P}_a$. In other words, $\mathcal{P}_b$'s history impacts the instantaneous probability of one event occurring at t for process $\mathcal{P}_a$.

The literature commonly exploits two class of processes for modeling NPPs: Hawkes [19] and Wold [40] processes. In both classes, one may re-write the intensity as a sum of two factors, an exogenous (Poissonian) constant, μ_a, and an endogenous factor accounting for $\mathcal{H}_\mathcal{P}(t)$: $\lambda_a(t|\mathcal{H}_a(t)) = \mu_a + \sum_{b=1}^{K} \alpha_{ba}\phi_a(\mathcal{H}_b(t))$, where ϕ is some influence kernel. Following this definition, let $\hat{\boldsymbol{A}} = [\alpha_{ba}]$ be a K by K matrix capturing the network structure of the processes, with $\alpha_{ba} \geq 0$. Here, $\alpha_{ba} = 0$ when $\mathcal{P}_b$ does **not** influence $\mathcal{P}_a$ (i.e., $\alpha_{ba}\phi_a(\mathcal{H}_b(t)) = 0$). Conversely, $\mathcal{P}_b$ **does** influence $\mathcal{P}_a$ when $\alpha_{ba} \geq 0$. In other words, an edge exists in the latent graph when $\alpha_{ba} \neq 0$. Inferring $\hat{\boldsymbol{A}}$ is done using timestamps (or in some cases cascades) only.

3.2 Models

We evaluated six different state-of-the-art models, chosen to represent both recent and older methods (2011 to 2018), with different characteristics (e.g., employ Hawkes, cascades and Wold Processes), and open-source implementations available online[1]. We also evaluate a classic Hawkes process with an Exponential kernel [19]. The runtime complexity per learning iteration for each model is measured according to K and N.

The most commonly employed Hawkes model is one with an Exponential influence function [19], simply called **ExpKern**. We learn this model using the open source `tick` library. It has an asymptotic cost of $O(NK^2)$ (K^2 parameters updated for each N timestamp per learning iteration). Out of the recent methods, from 2011 onward, the earliest we evaluate is HawkesEM (**HkEM**) [25]. It is a non-parametric estimator for the Hawkes processes with a complexity lower bound of $\Omega(NK^2)$ per iteration[2], where $\Omega(.)$

[1] `https://github.com/X-DataInitiative/tick`
`http://github.com/flaviovdf/granger-busca`
`http://snap.stanford.edu/netinf/`

[2] The asymptotic upper bound was not reported by the authors.

is a lower bound. In 2013, Zhou et al. proposed **ADM4** [45], a parametric Hawkes model that employs an exponential kernel with a fixed decay and complexity of $O(N^3K^2)$. We also tested **MLE-SGP** [41], which uses a parametrization of the kernel as a sum of M basis functions (Gaussian functions in the author's experiments) with a complexity of $O(MN^3K^2)$, where M is the number of basis functions. More recently, Achab et al. proposed HawkesCumulants (**HC**) [1], a non-parametric model that uses a moment matching method to compute the causal matrix. Because of this, the method avoids estimating the kernels themselves and the per-iteration complexity does not depend on N, but only on K. The per iteration complexity is $O(K^3)$.

For Wold processes only one option was available, namely Granger-Busca (**GB**) [15]. The method is an EM algorithm with complexity of $O(N(\log(N) + \log(K)))$. For the cascades framework, we tested **NetInf**[3], one of the state-of-the-art models. Cascade models are usually employed in datasets different than the ones we explore in this work (i.e., require cascade information), so we chose to evaluate **NetInf** on the only dataset where this is possible (Memetracker). The authors of this model state that there is no closed form for run-time complexity as it depends on the underlying network structure.

Hyper-parameters were optimized via random search [4]. As we discuss in the next section, when extracting networks via point processes, cross-validation is not usually employed [1, 15, 45]. A ground truth matrix of source destination pairs is constructed and models are learned using only events from either sources or destinations. Thus, the most appropriate model for the dataset is selected using the log-likelihood function. When available in the source code, we present results for the learned models with the best log-likelihood. For **HC**, the objective function [1] was used. Thus, we were able to properly optimize hyper-parameters for **HkEM**, **ADM4**, **ExpKern**, **GB** and for **HC**. The remaining method (**MLE-SGP**) does not have a log-likelihood function implemented. Thus, for this approach we keep the default values. Table 1 summarizes our models and hyper-parameters choices.

3.3 Ground Truth Data

In order to evaluate the models, we need a ground truth causal matrix $\boldsymbol{A}$. A common approach to construct this matrix is through datasets composed by $(source, destination, timestamp)$ triples, which indicates that some interaction (e.g. messages) occurred from $source$ to $destination$ at time $timestamp$. The ground truth matrix is, thus, captured by the normalized number of interactions between $(source, destination)$ pairs: $\boldsymbol{A}_{ij} = \#\text{events}(i \to j)/\#\text{events where } i \text{ is a source}$. To train the models on such triples, we remove the source column. Each process consists of destination nodes and their timestamps only. The models in these settings capture the causal notion that a received message can trigger other messages.

The most common dataset explored by these types of models is Memetracker [23], composed of publication times of articles across multiple web-domains. We selected the top 25 domains with the highest number of hyperlinks during the month of January, 2009. Sources are webpages, and the ground truth edges arise when the source creates

[3] Other models considered were MMEL [46] and Hawkes Conditional Law [2]. However, the first crashed consistently and the latter did not finish its execution on time.

Dataset	# of processes	# of events
CollegeMsg	100	17750
sx-askubuntu	28	3335
sx-mathoverflow	29	2767
sx-superuser	33	4157
email-Eu-core	100	11220
wiki-talk	100	30442
Memetracker	25	177163

Table 2. Number of processes and events for each dataset.

a hyperlink to a destination. Thus, the number of times the source domain cites the destination domain defines the whole matrix. Six other datasets were gathered from the Snap Network Repository [24] representing human communications in social networks, where sources and destinations are defined when one user contacts another. For each network, we selected the month with the highest number of events and only considered processes with more than 60 events each. If there were more than 100 processes that fulfilled this condition, we chose the top 100. Table 2 shows the number of processes and events for each dataset.

3.4 Metrics

We now describe the metrics used to evaluate the quality of the influence matrix inferred by the model. Let $\boldsymbol{A}$ be the $K \times K$ *ground truth* matrix and $\boldsymbol{A}_i$ the i_{th} row of this matrix. Similarly, we denote by $\hat{\boldsymbol{A}}$ the matrix generated by a model. We defined our metrics considering the rows of $\boldsymbol{A}_i$ and $\hat{\boldsymbol{A}}_i$, i.e., they capture the efficacy of the method from the viewpoint of a source node, or process $\mathcal{P}_i$.

Average Precision at n (AP@n). Initially, we calculate the Precision@n ($P@n$) as: $P@n(\boldsymbol{A}_i, \hat{\boldsymbol{A}}_i) = |\mathbb{T}_n(\boldsymbol{A}_i) \cap \mathbb{T}_n(\hat{\boldsymbol{A}}_i)|/n$, used in [15]. Here, $\mathbb{T}_n(\boldsymbol{A}_i)$ are the top n elements in $\boldsymbol{A}_i$ ordered by their value. The $P@n$ metric avoids the problem of sparsity, as it only considers the edges with the highest weight. However, it ignores the distribution of weights for the edges that are not in the top n. Average Precision at n ($AP@n$) aggregates the Precision at n for every possible n. By doing so, this metric solves the issue of choosing a specific n. However, equal weight is given to all of these choices. The NDCG, our next metric, mitigates this issue.

Normalized Discounted Cumulative Gain (NDCG). NDCG [11] is a measure of ranking quality that penalizes the errors according to the ranking. It is defined as $\frac{DCG_i}{IDCG_i}$, where for each row, $DCG_i(\boldsymbol{A}_i, \hat{\boldsymbol{A}}_i) = \sum_{\boldsymbol{A}_{ij} \in \boldsymbol{A}_i} \frac{2^{\boldsymbol{A}_{ij}}-1}{\log_2(pos^m(j)+1)}$ and $IDCG_i(\boldsymbol{A}_i, \hat{\boldsymbol{A}}_i) = \sum_{\boldsymbol{A}_{ij} \in \boldsymbol{A}_i} \frac{2^{\boldsymbol{A}_{ij}}-1}{\log_2(pos(j)+1)}$. $pos(j)$ captures the position of $\boldsymbol{A}_{ij}$ in the ranking of cell values $\boldsymbol{A}_{ij} \in \boldsymbol{A}_i$ and, similarly, $pos^m(j)$ is the position of $\hat{\boldsymbol{A}}_{ij}$ in the ranking of $\hat{\boldsymbol{A}}_{ij} \in \hat{\boldsymbol{A}}_i$. However, the NDCG does not penalize an overestimation of edge values if they are not too far up in the ranking. The NDCG varies from 0 to 1, the higher the better.

Normalized Root-mean-square error (NRMSE). NRMSE considers the difference in values between two matrices, being defined as the root-mean-square error between the cells of the estimated and ground truth matrices, normalized by the range of values: $NRMSE(\boldsymbol{A}_i, \hat{\boldsymbol{A}}_i) = \frac{\sqrt{\sum_i (\boldsymbol{A}_{ij} - \hat{\boldsymbol{A}}_{ij})^2/K}}{\boldsymbol{A}_{imax} - \boldsymbol{A}_{imin}}$, where $\boldsymbol{A}_{imax}$ and $\boldsymbol{A}_{imin}$ are the maximum and minimum values for the vector $\boldsymbol{A}_i$, respectively.

Other metrics were used to evaluate NPPs, such as the average row rank correlation coefficient [1, 15, 45] and the relative error [1, 15, 41, 45]. However, these metrics suffer from small sample sizes (non-zero columns in each rows) due to graph sparsity. It is expected that small samples lead to less statistical significance, thus we shall limit our discussion to the subset of complementary metrics that are able to reject a higher number null-hypothesis tests (e.g., we have a higher confidence that their values did not arise due to chance).

3.5 Null Models and Confidence Intervals

We also use a Null model composed of a random generation (or permutation) of rankings for the ranking metrics (e.g., AP@n). For the remaining metrics, we generate a random permutation of the values of the ground truth matrix. This matrix allows the null model to maintain the sparsity and usual range of values of the original ground truth matrix. These permutations are generated 1000 times and compared to the ground truth matrix using the metrics to create a confidence interval of 95% for each (metric, dataset) pair.

We also generated confidence intervals for the estimates calculated with the real datasets. As we do not know the true stochastic data generating mechanism, we used a bootstrap procedure adapted to point processes data [26, 34]. The time axis $(0, T]$ was partitioned into 10 parts of equal length: $\mathcal{T} = (0, T/10], \ldots, (9T/10, T]$ and new pseudo-datasets were created by gluing together 10 temporal pieces sampling with replacements from $\mathcal{T}$. The estimates were calculated in the pseudo datasets, and this procedure was repeated 100 times to generate the sampling distributions from which confidence intervals were derived. In this paper, we present the first comparison of models under statistical significance tests.

3.6 Upper Bounds and Optimization

As with any ML task, an important step when comparing models is testing and choosing hyper-parameters. However, in the setting of Latent Network Extraction, we do not have validation sets to optimize hyper-parameters. That is, starting from an intensity function $\lambda_a(t)$ (see Section 3.1), we derive a likelihood for the entire dataset [12]. This is one of the most important results from Point Processes, and the intensity is a sufficient function for learning model parameters. Using the theory of Maximum Likelihood Estimation (MLE), we can argue that hyper-parameters should be chosen to maximize the likelihood for the datasets. However, which guarantees do we have that this model in-fact performs the best network extraction? Notice that without any edge, we cannot test the hyper-parameters on the validation data. It turns out that (from our empirical results in the next section) most methods do not present a correlation of likelihood with the aforementioned efficacy metrics. We now detail how we used the expected validation [13] alongside with this correlation to evaluate the NPP models.

Given a choice of n hyper-parameter configurations, we can compute the expected validation score (for a choice of evaluation metric) of the models by assuming that the graph is known. That is, we may simply pick the choice of hyper-parameters that maximizes some accuracy metric and not the likelihood. This score is captured via the

equation:

$$\mathbb{E}[V_n^*|n] = \sum_{\upsilon} \upsilon \; P(V_n^* = \upsilon|n) = \sum_{\upsilon} \upsilon \; (\hat{P}(V_i \leq \upsilon)^n - \hat{P}(V_i < \upsilon)^n).$$

Here, V_n^* is the maximum score after testing n hyper-parameters configurations, υ are the observed scores and V_i is the score for the i-th configuration, drawn i.i.d. (with random search, for example). Given that hyper-parameter configurations are i.i.d., $\hat{P}(V_i \leq \upsilon)^n$ captures the probability of observing n configurations below or equal to the threshold υ. Similarly, $\hat{P}(V_i < \upsilon)^n$ captures the probability of these n configurations being below υ. By iterating over all observed values of the given score υ, this function is thus an expectation of the score given that: $\hat{P}(V_i = \upsilon) = \hat{P}(V_i \leq \upsilon)^n - \hat{P}(V_i < \upsilon)^n$. Thus this value is simply an expected value of the score when testing n hyper-parameter configurations.

Conditioning on n is interesting as it measures the expected validation score as a function of the computational power set forth to optimize the model. More hyper-parameter tests imply a need of more computational resources. Thus, from this metric we can infer not only an **upper-bound** on the score, as the graph is known, but also have some notion of the expected computing cycles needed to achieve such a score. After computing the value of $\mathbb{E}[V_n^*|n]$, we analyze if there is a correlation between a higher likelihood and a higher value for the efficacy metrics. If a correlation is present, then we can argue that such an upper-bound is achievable by the method. If not, we can state that optimizing the likelihood will not lead to better results for that model. To have some notion of the variability of this estimate, we can also compute the variance $\mathbb{V}[V_n^*|n] = \mathbb{E}[V_n^{2*}|n] - \mathbb{E}[V_n^*|n]^2$ by computing the the expected value using υ^2 in order to estimate $\mathbb{E}[V_n^{2*}|n]$.

3.7 Sensitivity to False Positives

When processes are known to be independent among themselves, LNE models must yield a zero matrix $\hat{\boldsymbol{A}}$ and we can test them for *false positives*. To do so, we simulate $K = 10$ independent Poisson (homogeneous and inhomogeneous) processes with $n = 1000$ events each. A Poisson process is independent of its history, and its intensity function is given by $\lambda(t|\mathcal{H}_a(t)) = \lambda(t)$. In this case, besides comparing $\hat{\boldsymbol{A}}$ with a zero matrix, the models should also infer the *baseline intensities* μ, which should be similar to the average $\lambda(t)$, or simply n/T.

4 Results

We start this section with a discussion about the empirical maximum values each model can achieve. Next, we evaluate the models' resilience to false positives using synthetic data. Experiments were run on an i7-6700 CPU with eight cores and 32GB of RAM.

4.1 Upper Bounds and Optimization

Three factors are considered for assessing the accuracy of the models when learning the network structure:

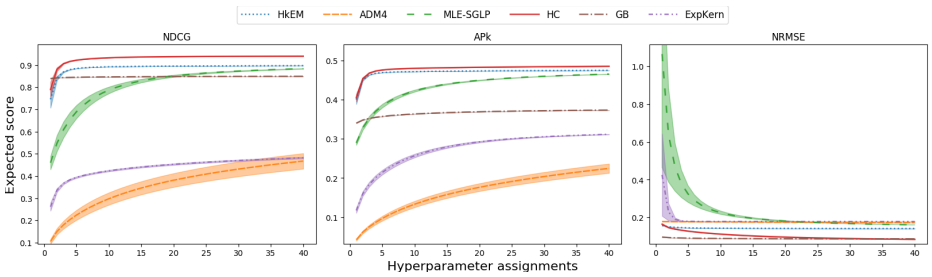

Fig. 1. Expected value of the best score according to the number of hyper-parameter configurations tested for the *sx-superuser* dataset.

1. **F1:** the best score achieved by the model when it is tuned specifically to improve the comparison metric over the ground truth network;
2. **F2:** the number of hyper-parameter configurations needed to be tested to achieve this maximum score;
3. **F3:** whether optimizing the model without using the ground truth network (i.e. using the likelihood or loss function over the timestamps) can improve the model's performance measured by the comparison metric.

These analyses serve to verify whether the models are well-defined, i.e., if optimizing the likelihood leads to better estimates of $\boldsymbol{A}$. Hyper-parameter optimization is done via random-search. Each step of the search defines a hyper-parameter configuration. For every dataset and hyper-parameter configuration, we computed the expected validation score $\mathbb{E}[V_n^*|n]$ and the variance $\mathbb{V}[V_n^*|n]$, which may vary according with the number of hyper-parameter configurations tested. In Figure 1, we show the expected score for the models on the *sx-superuser* dataset as a function of the number of hyper-parameter configurations tested. The variance is shown as the shaded region. For each curve we can derive two values: the number of hyper-parameter tests necessary until the metric no longer improves (defined as an increase of less than 0.001) and the metric value at this step, which we refer as the **upper-bound** of the triple (method, dataset, metric).

In Figure 2 we plot these two quantities for every triple (method, dataset, metric). We do not present results for **MLE-SGP** on the Memetracker dataset because it did not converge. Each method is determined by a color and each dataset by a symbol. The shaded region on the plots are determined by median values on the x and y-axis. When inside this region, we can state that the method and dataset pair is above the overall median for the corresponding axis considering the given metric. In the caption of the figure we describe how many times each method fell inside this region. Higher values of counts indicates that the method is performing well for (dataset, metric) pairs. In total, we have 7 datasets and 3 metrics (21 results), so the ideal method would be counted 21 times. In general, **HkEM**, **HC** and **GB** obtained the best upper bounds with few hyper-parameter tests, and the remaining methods need a larger amount of tests and obtain varying results. **HkEM** and **GB**, both with 13 wins, require few hyper-parameter tests to reach their upper-bound.

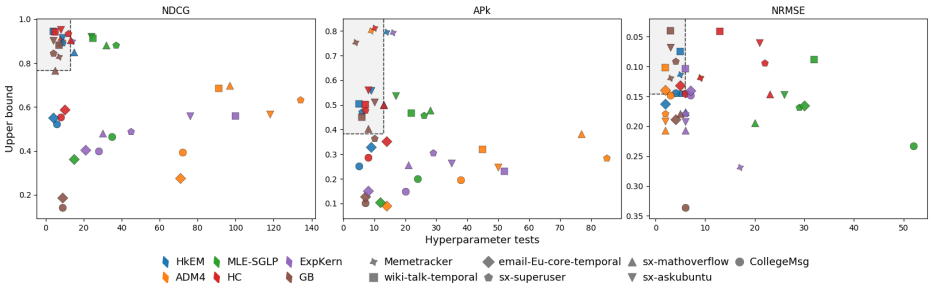

Fig. 2. Maximum expected value achieved vs number of hyper-parameter tests. The shaded area is where methods are above the median for both axis. Counts for how many times each model was in the shaded area: **HkEM**: 13, **ADM4**: 4, **MLE-SGP**: 0, **HC**: 12, **GB**: 13, **ExpKern**: 1.

Recall that a model can only approximate the upper bound in practice if optimizing its likelihood (or loss function) improves the quality of the recovered network (**F1**). Thus, in Figure 3 we show the Kendall-tau correlation (τ) between the value of the metric and the likelihood of each model when this value is achieved. Every point is a (metric value, likelihood) pair and each sub-figure refers to a (method, metric) pair. Due to space limitation, we only show these results for the *sx-superuser* dataset. Notice that only **HkEM** and **ADM4** have high correlations regardless of the evaluation metric. This result is an initial indicator that these methods are better defined than the others. Being ill-defined means that optimizing the likelihood will not lead to better network recovery.

In Figure 4, we summarize these results for all datasets by plotting the Kendall-tau correlation and the upper-bound for the models for all metrics and datasets. Note that **MLE-SGP** is not included in this analysis since no implemented likelihood function was available for testing. Similar to Figure 2, we also present a shaded region indicating where models out-perform others (medians of the x and y-axis). Observe that **HkEM** tends to have high correlation values and usually outperforms others (11 wins), being followed by **ADM4** (7 wins). **GB** (2 wins), which was a good candidate before, regularly shows no correlation. The remaining models also performed poorly.

This second analysis argues in favor of **HkEM** and **ADM4**. Our results so far are based on a hypothetical setting in which we are able to measure validation scores. Some methods (e.g. **HC**) appear to be able to reach high metric values (e.g. NDCG) but the lack of correlation limits their applicability. To provide a more accurate evaluation, we next discuss the real-world setting where such optimization is not possible, that is, there is no ground truth and only the likelihood is available for hyper-parameter tuning.

4.2 Experimental Results for the LNE task

In Figure 5 we show the overall results of the models for the latent network extraction (LNE) task. Each plot contains the results for a given metric. The markers are the values models achieved when executed on the original timestamp data and the lines are the 95% confidence intervals (computed using bootstrap). Areas where the models are equivalent or worse than the Null model (95% confidence interval) are shaded in gray. In a few cases, the result on the original data lies marginally outside of the confidence interval.

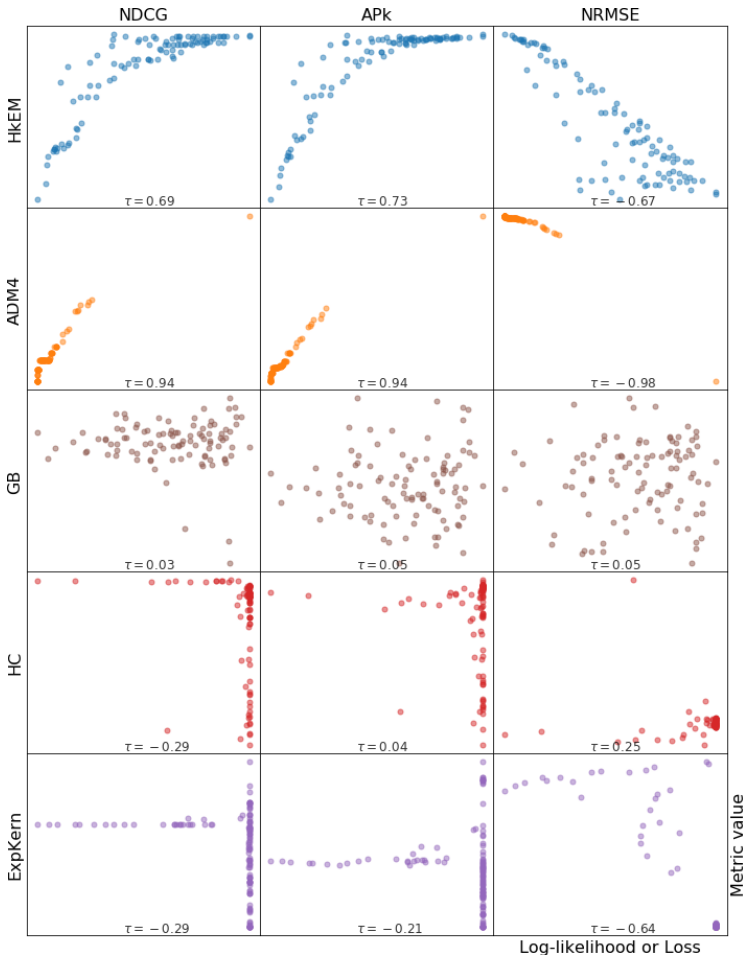

Fig. 3. Log-likelihood of the model (or loss function for **HC**) vs metric values when comparing the generated network with the ground truth for the *sx-superuser* dataset.

This is expected because Web datasets are bursty, and results may depend on specific periods of time left out by the bootstrap. Hyper-parameters were optimized using the log-likelihood or loss function. For **MLE-SGP**, the default parameters were used.

Again, **HkEM** achieved the best results, being the best model in 9 out of the 21 combinations of dataset and metric, never overlapping with the null model's range. **ADM4**, **MLE-SGP**, **HC** and **GB** followed next, with each one getting the best value for 3 (dataset, metric) pairs. **ExpKern** was never the best model. These results can be well explained by the previous experiments. A model with good correlation and upper bound managed to get the best scores (**HkEM**), while models without a good correlation are difficult to optimize and tend to have erratic behaviour (**HC** and **ExpKern**).

Considering the bootstrap intervals, **HkEM** was tied for the best model 14 times. **MLE-SGP** and **GB** were the best models 12 times, **HC** 8 times, **ADM4** 5 times, and

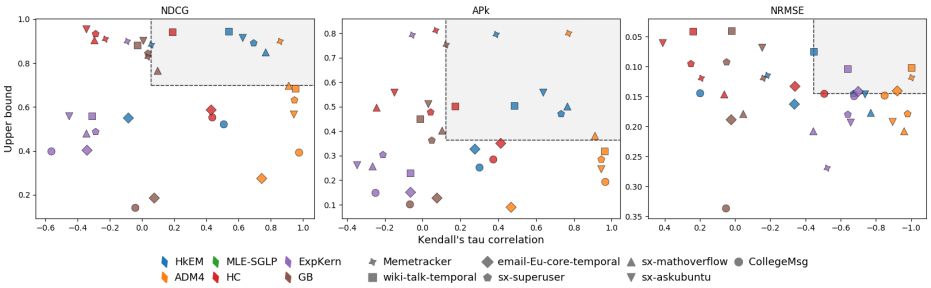

Fig. 4. Maximum expected value achieved by a model versus correlation between likelihood and metrics. The shaded area is where methods are above the median for both axis. Counts for how many times each model was in the shaded area are **HkEM**: 11, **ADM4**: 7, **GB**: 2, **HC**: 3, **ExpKern**: 2.

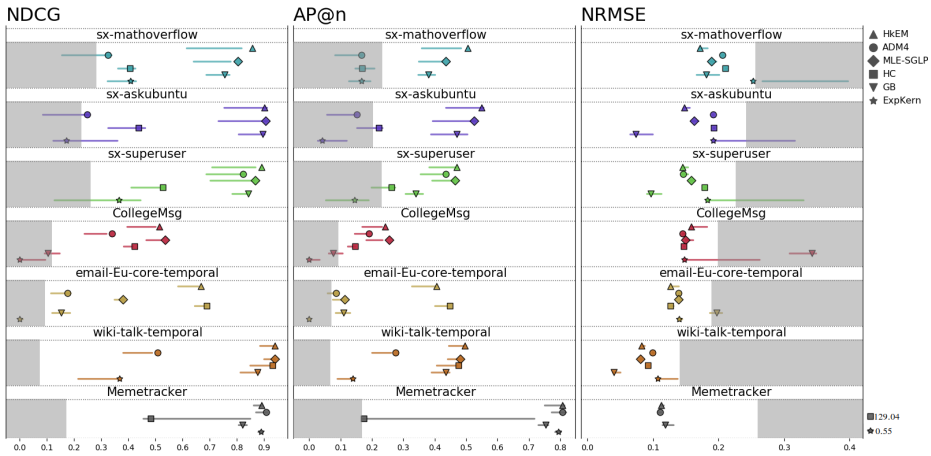

Fig. 5. Evaluations for every method in all datasets. Each marker is the result of the model in the original dataset and the lines are 95% confidence intervals calculated through bootstrap. The shaded regions correspond to the confidence interval of null models.

ExpKern 2 times. Concerning the datasets, Memetracker seems to be the easiest to capture the network structure. Relationships defined from links created from one process to another may be more suitable for NPP models than when defined from message exchanges. On the other hand, datasets without self-excitation (*CollegeMsg* and *email-Eu-core*) are the hardest to infer the latent network.

Our results so far argue in favor of **HkEM** and possibly **ADM4** as the best methods. Nevertheless, we point out that **GB** should not be ignored as it is the only method that executes on Web scale data [15] (see asymptotic cost in Section 3). Also, **HC** tends to have very high upper-bound values, but the lack of correlation leads the model to under-perform when used for the LNE task. This is not the case for **GB** as its efficacy appears to be less sensitive to the choice of hyper-parameters.

Method/Metric	Homogeneous		Inhomogeneous	
	Baseline error	**LN error**	**Baseline error**	**LN error**
HkEM	1.00(1.00, 1.00)	1.08(1.06, 1.10)	1.00(1.00, 1.00)	**3.47(3.27, 3.67)**
ADM4	**0.02(0.01, 0.03)**	**0.05(0.03, 0.07)**	0.71(0.68, 0.76)	6.03(5.86, 6.16)
MLE-SGP	0.39(0.30, 0.48)	1.61(1.14, 2.12)	0.92(0.89, 0.97)	20.11(19.45, 20.79)
HC	2.44(0.75, 37.32)	22.46(14.48, 729.34)	2.46(1.28, 3.60)	8.63(5.47, 34.30)
GB	0.32(0.31, 0.34)	-	**0.40(0.39, 0.41)**	-
ExpKern	0.03(0.02, 0.05)	0.07(0.05, 0.09)	0.68(0.65, 0.70)	4.50(4.32, 4.67)

Table 3. MSE for the baselines and Granger matrix on simulated Poisson processes.

4.3 Comparison with cascades method

We now compare the methods when more information (e.g., cascade sets) are available for usage. In particular, we compare Hawkes and Wold models with **NetInf** [32], the only cascades model. Because **NetInf** requires cascade data, our analysis is limited for the Memetracker dataset. Also, since the problem formulation is slightly different, we processed the Memetrackler data in two different forms. In Memetracker, cascades are captured when short sentences are repeated over websites. Thus, to determine cascades, sentences with more than 3 words were extracted together with the timestamps when each different website first mentioned the sentence. The sequence of timestamps of a sentence form a cascade. While all the other models process only the sequence of timestamps on each website, **NetInf** also processes the cascade to which each timestamp belong. Only the top $K = 50$ websites in terms of hyperlink mentions were considered in our analysis. The ground truth is the hyperlink counts as described in Section 3.3, a total of 182,379 events.

It is also important to note that **NetInf** infers an unweighted network, thus the metrics that employ rankings (AP@n and NDCG) become meaningless. For this reason, we limit our comparison using NRSME. We found that **NetInf** achieves a worse score than most of the other Hawkes and Wold models, with a NRMSE of of 0.398. Out of the other six methods, only **ExpKern** had a higher value (0.616). This result is quite remarkable as the Hawkes and Wold methods do not explore cascades. That is, they make use of less information than **NetInf**. Overall, these results point that exploring cascade data may not be a necessity in real world settings. Full results are presented in the Supplementary Information material [6].

4.4 False Positive Analysis (Simulations)

To test for false positives, we simulated $K = 10$ **homogeneous** Poisson processes with $\lambda(t) = 1$ for $N = 10,000$ steps. In this case, all models should infer baseline (or background) intensities $\mu_a = 1$ for each process a and $\hat{\boldsymbol{A}}_{ij} = 0\ \forall i, j$. Next, we simulate 10 **inhomogeneous** Poisson processes, where each process indexed by $a = \{1, 2, \cdots, 10\}$ has intensity $\lambda_a(t) = 1 + \sin(\omega_a t)$, where $\omega_a = \frac{1}{2a} \in \mathbb{R}$. For both experiments we run the models over the simulated timestamps and repeated this process 200 times to calculate the confidence intervals.

Table 3 shows the mean squared error (MSE) of the estimated baseline intensities (μ_a) with respect to the Poisson intensities ($\lambda(t)$), denoted by **baseline error**, and the

MSE of the cells of the matrix with respect to the all zero matrix, denoted by **LN error**. The values between parenthesis are the bounds for the 95% confidence interval. Note that we do not calculate the squared error of the matrix estimated by **GB**, because it is a row stochastic matrix and, thus, would never generate a zeroed row.

Surprisingly, only **ADM4** and **ExpKern** were able to capture the *homogeneous* Poisson processes successfully, with both errors close to 0. The probable reason is that both models impose an exponential kernel on its intensity function, while all the other Hawkes methods try to approximate a kernel function, which can lead to overfitting. **GB** performed fairly well in estimating the baseline intensities, having the third smallest error, followed by **HkEM** and **MLE-SGP**. However, part of the events of these three models were still estimated to be caused by other processes. **HC** was the one with the worst performance, having the highest errors for the baseline and the matrix. Regarding the *inhomogeneous* Poisson processes, all models identified a dependence structure among the processes, which does not exist. **GB**, **ADM4** and **ExpKern** models had the smallest baseline errors. **HkEM** had the closest estimate for the network, followed by **ExpKern** and **ADM4**. Other Hawkes models had higher errors.

5 Conclusions

Laborious, rigorous evaluations and scrutiny is a necessity in any scientific field. As we have argued throughout this paper, the machine learning (ML) field is currently at a position where there is a need to revisit its state of the art, given the plethora of applications and scenarios involved. In particular, empirical rigor is critical to avoid spurious findings and to delineate the scope of ML models. In his original review of the "two cultures" [8], Breiman made an in-depth comparison between theory-based models and empirical prediction-based algorithms, concluding that researchers must move away from exclusive dependence on data models and adopt a more diverse set of tools. Similarly, machine learning researchers [20, 22, 27, 30, 35–38] are taking a step back and asking themselves if too much effort is put on sophisticated and complex ideas that, in practice, perform worse than traditional approaches.

This work serves as a methodological guide to new studies on network inference from networked point processes and corroborates with the warnings mentioned above, which asks for more empirical rigor in ML research. Based on our results, we argue that the classical **HkEM** model is the best general-purpose NPP model as it had superior results in most settings and positive correlations for likelihood and evaluation scores. **ADM4**, while generally less useful than **HkEM**, also presents strong positive correlations and is the best model for detecting false positives. Although **GB** does not present a correlation between likelihood and efficacy, it is the only method known to scale for full Web datasets [15], being recommended for these cases. The other models have not excelled in any of the five evaluation fronts.

Acknowledgements

Funding was provided by the authors' grants from CNPq, CAPES, and Fapemig. This project was also partially funded via CNPq's Universal Grant 421884/2018-5.

References

1. M. Achab, E. Bacry, S. Gaiffas, I. Mastromatteo, and J.-F. Muzy. Uncovering causality from multivariate hawkes integrated cumulants. In *ICML*, 2017.
2. E. Bacry and J.-F. Muzy. Second order statistics characterization of hawkes processes and non-parametric estimation. *arXiv preprint arXiv:1401.0903*, 2014.
3. A.-L. Barabási, R. Albert, and H. Jeong. Scale-free characteristics of random networks: the topology of the world-wide web. *Physica A: statistical mechanics and its applications*, 281(1-4):69–77, 2000.
4. J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.
5. C. Blundell, J. Beck, and K. A. Heller. Modelling reciprocating relationships with hawkes processes. In *NeuRIPS*, pages 2600–2608, 2012.
6. G. R. Borges, F. Figueiredo, R. Assunção, and P. O. Vaz-de Melo. Networked point process models under the lens of scrutiny: Supplementary information material. `https://github.com/guilhermeresende/NPPs/`, 2020. [Online; accessed 24-June-2020].
7. S. Bourigault, S. Lamprier, and P. Gallinari. Representation learning for information diffusion through social networks: an embedded cascade model. In *WWW*, 2016.
8. L. Breiman. Statistical modeling: The two cultures. *Statistical science*, 16(3), 2001.
9. S. Chen, A. Shojaie, E. Shea-Brown, and D. Witten. The multivariate hawkes process in high dimensions: Beyond mutual excitation. *arXiv preprint arXiv:1707.04928*, 2017.
10. E. Choi, N. Du, R. Chen, L. Song, and J. Sun. Constructing disease network and temporal progression model via context-sensitive hawkes process. In *ICDM*, 2015.
11. W. B. Croft, D. Metzler, and T. Strohman. *Search engines: Information retrieval in practice*, volume 520. Addison-Wesley Reading, 2010.
12. D. J. Daley and D. Vere-Jones. *An introduction to the theory of point processes, vol. 1*. Springer, New York, 2003.
13. J. Dodge, S. Gururangan, D. Card, R. Schwartz, and N. A. Smith. Show your work: Improved reporting of experimental results. In *EMNLP*, 2019.
14. M. Eichler, R. Dahlhaus, and J. Dueck. Graphical modeling for multivariate hawkes processes with nonparametric link functions. *Journal of Time Series Analysis*, 38(2):225–242, 2017.
15. F. Figueiredo, G. R. Borges, P. O. Vaz-de Melo, and R. Assunção. Fast estimation of causal interactions using wold processes. In *NeuRIPS*, pages 2975–2986, 2018.
16. E. Ghalebi, B. Mirzasoleiman, R. Grosu, and J. Leskovec. Dynamic network model from partial observations. In *NeuRIPS*, pages 9862–9872, 2018.
17. M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. *ACM Trans. Knowl. Discov. Data*, 5(4), Feb. 2012.
18. M. Gomez Rodriguez, J. Leskovec, and B. Schölkopf. Structure and dynamics of information pathways in online media. In *WSDM*, 2013.
19. A. G. Hawkes. Point spectra of some mutually exciting point processes. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 438–443, 1971.
20. M. Jie, G. S. Collins, E. W. Steyerberg, J. Y. Verbakel, B. van Calster, et al. A systematic review shows no performance benefit of machine learning over logistic regression for clinical prediction models. *Journal of clinical epidemiology*, 2019.
21. R. Junuthula, M. Haghdan, K. S. Xu, and V. Devabhaktuni. The block point process model for continuous-time event-based dynamic networks. In *WWW*, 2019.
22. W. Kryscinski, N. S. Keskar, B. McCann, C. Xiong, and R. Socher. Neural Text Summarization: A Critical Evaluation. In *EMNLP-IJCNLP*, 2019.
23. J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD*, 2009.

24. J. Leskovec and A. Krevl. SNAP Datasets. `http://snap.stanford.edu/data`.
25. E. Lewis and G. Mohler. A nonparametric em algorithm for multiscale hawkes processes. *Journal of Nonparametric Statistics*, 1(1):1–20, 2011.
26. J. Loh and M. Stein. Bootstrapping a spatial point process. *Statistica Sinica*, 14(1), 2004.
27. M. Lucic, K. Kurach, M. Michalski, O. Bousquet, and S. Gelly. Are Gans created equal? A large-scale study. In *NeuRIPS*, 2018.
28. C. Mavroforakis, I. Valera, and M. Gomez-Rodriguez. Modeling the dynamics of learning activity on the web. In *WWW*, 2017.
29. H. Mei and J. M. Eisner. The neural hawkes process: A neurally self-modulating multivariate point process. In *NeuRIPS*, 2017.
30. G. Melis, C. Dyer, and P. Blunsom. On the state of the art of evaluation in neural language models. In *ICLR*, 2018.
31. M. A. Rizoiu, L. Xie, S. Sanner, M. Cebrian, H. Yu, and P. Van Hentenryck. Expecting to be hip: Hawkes intensity processes for social media popularity. In *WWW*, 2017.
32. M. G. Rodriguez, D. Balduzzi, and B. Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *ICML*, 2011.
33. T. Santos, S. Walk, R. Kern, M. Strohmaier, and D. Helic. Self-and cross-excitation in stack exchange question & answer communities. In *WWW*, 2019.
34. S. V. Sarma, D. P. Nguyen, G. Czanner, S. Wirth, M. A. Wilson, W. Suzuki, and E. N. Brown. Computing confidence intervals for point process models. *Neural computation*, 23(11):2731–2745, 2011.
35. D. Sculley, J. Snoek, A. Rahimi, and A. Wiltschko. Winner's curse? on pace, progress, and empirical rigor. In *ICLR*, 2018.
36. B. Strang, P. van der Putten, J. N. van Rijn, and F. Hutter. Don't Rule Out Simple Models Prematurely: A Large Scale Benchmark Comparing Linear and Non-linear Classifiers in OpenML. In *IDA*, 2018.
37. G. Tucker, J. Snoek, C. Riquelme, G. Tucker, and J. Snoek. Deep Bayesian Bandits Showdown: An Empirical Comparison of Bayesian Deep Networks for Thompson Sampling. In *ICLR*, 2018.
38. A. Vaswani, G. Brain, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention Is All You Need. *NeuRIPS*, 2017.
39. J. Wang, V. W. Zheng, Z. Liu, and K. C.-C. Chang. Topological recurrent neural network for diffusion prediction. In *ICDM*, 2017.
40. H. Wold. On stationary point processes and markov chains. *Scandinavian Actuarial Journal*, 1948(1-2):229–240, 1948.
41. H. Xu, M. Farajtabar, and H. Zha. Learning granger causality for hawkes processes. In *ICML*, 2016.
42. W. Yang, K. Lu, P. Yang, and J. Lin. Critically examining the "neural hype": Weak baselines and the additivity of effectiveness gains from neural ranking model. In *SIGIR*, 2019.
43. Y. Yang, J. Etesami, N. He, and N. Kiyavash. Online learning for multivariate hawkes processes. In *NeuRIPS*, 2017.
44. L. Yi, B. Liu, and X. Li. Eliminating noisy information in web pages for data mining. In *KDD*, 2003.
45. K. Zhou, H. Zha, and L. Song. Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes. In *AISTATS*, 2013.
46. K. Zhou, H. Zha, and L. Song. Learning triggering kernels for multi-dimensional hawkes processes. In *ICML*, 2013.