Uncovering Social Links through Stochastic Point Processes

Rui Zhang (u5963436)
Research School of Computer Science
The Australian National University

April 2017

Abstract

Social networks can help to understand such as the jobs and hobbies of people and they are often approximated by retweet networks. However, retweet networks on the Twitter has a star-structure (Figure 2) which cannot reflect the real social networks. One of state-of-the-art methods, NETINF[3] is used to infer edges of retweet networks but it requires numerous retweet cascades to recover a retweet network. Sometimes, there is only a cascade occurring without more cascades for improving the accuracy. To address this problem, we use a Hawkes Point Process with the Power-law kernel to model retweet cascades and apply the branching structure of Hawkes Point Process to retrieve the real retweet network. Optimizing model parameters and inferring retweet networks are combined by Expectation Maximization (EM) algorithm. We validate our approach by inferring the social graph behind a dataset of Twitter cascades related to news. We use the Twitter crawler and the Twitter API to construct the dataset prepare the dataset for experiments by using the Twitter crawler to collect retweet cascades and downloading the friend lists of users in the tweet cascades by the Twitter API. We evaluate the performance of our method by conducting experiments on the synthetic and real data. In the experiments, we find that our method can optimize model parameters as well as Maximizing Likelihood Estimation[12] and it gets good performance on the real data by comparing with seven baselines.

1 Introduction

Nowadays, people tend to get news from the Internet such as the Twitter and Youtube rather than reading newspaper and watching TV. Some latest articles and comments are released on the popular social media platform, which are read, liked and retweeted to diffuse on the Internet and the diffusion paths compose a retweet network. For example, the retweet network shown in Figure 1 which has a tree structure and we can see how tweets diffuse and some users like the owner of the Tweet 3 play an important role in the diffusion because they cause lots of retweets and promote the diffusion. Thus, based on the retweet network, people can analyze how information diffuses and the importance of users in the diffusion.

The benefits of retweet networks motivate interest of people in it but some retweet network is unavailable such as the retweet network on the Twitter. For example, Figure 2 shows a retweet network constructed by data from the Twitter API corresponding to the real diffusion in the Figure 2. It has a star structure where all retweet are indicated to be from the Tweet 1 (the root tweet) and clearly does not reflect the real diffusion. For example, the Tweet 9 in the real retweet network(Figure 1) is a retweet of the Tweet 3 but the one in Figure 2 is indicated to be from the Tweet 1.

One of common methods to solve this problem is to design probability distribution of edges in the retweet networks in respect of the time and/or the number of users' followers. This kind of method predicts the potential retweet networks based on the probabilities of edges. The high probability reflects the high possibility of an edge being included in the retweet network. Besides, NETINF and a series of methods extending it get good performance in recovering retweet networks

[3, 4, 14]. Their approach is to select edges of retweet networks which can bring about the maximal improvement of the log-likelihood for all cascades occurring in a graph, which can be described as:

$$e_i = \underset{e \in G/G_{i-1}}{\operatorname{argmax}} F_C(G_{i-1} \cup \{e\}) - F_C(G_{i-1}) \tag{1}$$

$$F_C(G) = \sum_{c \in C} F_c(G) \tag{2}$$

where $F_c(G)$ is the improvement of log-likelihood for the cascade c occurring in graph G. Distribution-based methods and NETINF's all need cascades to optimize the parameters in their models and besides, NETINF's cannot get good performance unless there exist numerous retweet cascades occuring on the retweet network, which are needed to improve the prediction accuracy. Therefore, when there is only one cascade occurring in the retweet network and no more retweet cascades available for training or improving the prediction accuracy, both kinds of methods will not effectively recover the retweet networks.

In this project, we aim to infer retweet networks with only one cascade. We propose a method based on Hawkes Point Processes with the Power-law kernel[5]. The Hawkes Point Process model is well-known for its capacity of modeling the self-exciting characteristic of diffusion. The self-exciting features are common in, such as earthquakes and trade. A large earthquake tends to trigger aftershocks and a large sell order usually causes other sell orders. Likewise, a tweet or retweet of an influential is likely to result in plenty of retweets of people seeing it. Except the self-exciting feature, the branching structure in the retweet cascades are important for recovering the retweet network and they can be modeled by the branching structure of the Hawkes Point Process. For example in the Figure 3, there is a tweet (t_1, m_1) (root tweet) released by an influential. Note that in this report, a tweet is represented by its occurring time and the number of the owner's followers. The followers of the influential see, like and retweet the tweet and as a result, what people observe is an event sequence finally with hidden parenthood relations.

To infer the parenthood relations, we introduce p_{ij} to represent the probability of the parenthood relations. For example, we can use p_{32} as the probability of the Tweet 2 being the parent of the Tweet 3. Given p_{ij}, we can determine which parenthood relation is more probable. For example, we get $p_{32} = 0.3$ and $p_{31} = 0.7$ and we think the Tweet 3 is more likely to be retweeted from the Tweet 1. In our proposed method, we optimize the Hawkes Point Process model and p_{ij} simultaneously by EM algorithm. After modeling and optimization, we construct the Twitter dataset by the Twitter crawler and the Twitter API and evaluate our method on these real data as well as some synthetic data generated from the simulation of Hawkes Point Process.

The main contributions of this work are as follows:

- a Hawkes-Point-Process-based approach to infer true parenthood relations between tweets and the hidden social graph using single cascades.
- through comparison with 7 baselines and state-of-the-art algorithms.
- construction of new dataset of news diffusion.
- theoretical analysis of the applicability of different EM variants (convexity analysis).

Apart from the introduction to the background and our methods in this section, the section 2 is going to introduce related work to our project, mainly including models for the self-exciting feature in different kinds of data and branching structure of diffusion as well as optimization algorithms proposed for Hawkes Point Process models. The section 3 gives the details of our method, i.e., the concrete mathematical expression of Hawkes Point Process and the Power-law triggering kernel and how EM algorithm combines inferring hidden variables p_{ij} with optimizing model parameters. In the section 4, how to construct the Twitter dataset by the Twitter crawler and the Twitter API is explained. In the section 5, we introduce how we conduct experiments on the synthetic and real data and show and analyze the performance of our method. In the section 6, there is a brief summary of this project.

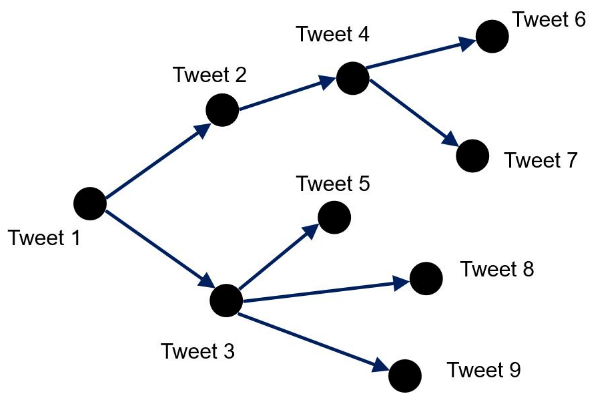

Figure 1: The real retweet network as a tree

The real retweet network reflects the real diffusion of tweets and it has a tree structure. Each edge represents the diffusion direction of a tweet.

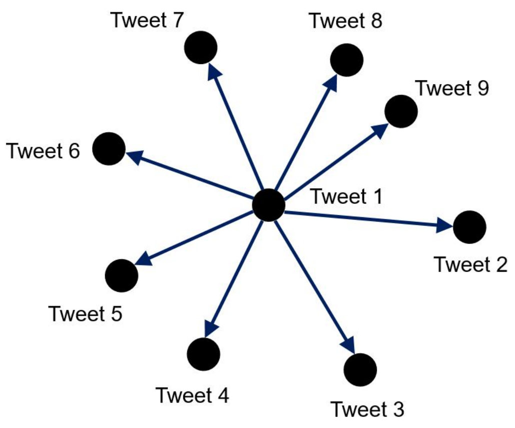

Figure 2: The retweet network constructed by the Twitter API

The network corresponds to the diffusion in Figure 1. The center point denotes the root tweet and peripheral points retweets. All retweets are indicated to be from the root tweet.

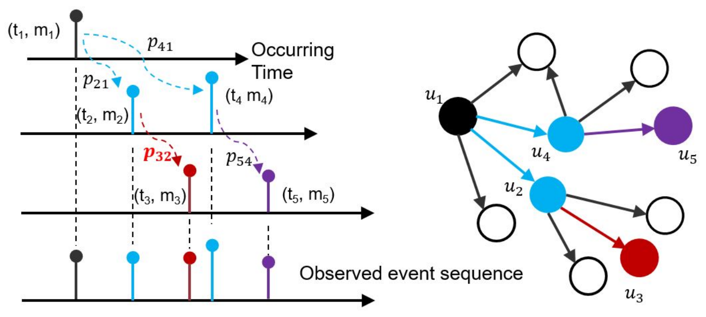

Figure 3: The branching structure of the retweeting process

The branching structure contains the parenthood relations between tweets and to get the most probable retweet network, we introduce p_{ij} to represent the probability of the Tweet i being from the Tweet j. Among $\{p_{ij}\}$, $j = 1, ..., i-1$, the edge with the higher probability is more likely to occur in the retweet network.

2 Related Work

This project focuses on modeling the self-exciting feature in the point processes and uncovering the diffusion structure. We introduce the related work to this project in 3 sections: (1) modeling diffusions with self-exciting processes, (2) uncovering hidden social graphs and (3) EM variants.

2.1 Modeling Diffusions with Self-exciting Processes

There have been lots of work trying to modeling the self-exciting feature of different data such as tweets[12], news articles [12], Youtube videos [13] and maintenance data of the components of a general infusion pump [16]. In [12], they use the Hawkes Point Process model with the Power-law triggering kernel in their method and consider various features including virality of tweet contents, user influence and memory decay in their triggering kernel. By using the L-BFGS algorithm to maximize the log-likelihood of cascades, they get optimal parameters of their model and predict the popularity of tweets and news articles. Tweets and news articles have a shared characteristic: the occurrence of a tweet or a new article is at a time point rather than roughly in a time interval. However, for the views of Youtube videos, we can only get the count of views in a specific time interval such as the last 24 hours so the time when a video is watched is unknown and Point-Process-based methods cannot effectively deal with these data any more. In order to deal with this problem, [13] propose event intensity, i.e., the expectation of the event rate over the event history. By minimizing the difference between event intensity of models and data, they train their model on views of videos in 90 days, which get a good prediction accuracy in predicting the popularity of these videos in the following 30 days.

Except modeling the self-exciting feature in the popular social media data, researchers also analyze the maintenance data in the engineering by modeling the self-exciting feature in these data[16]. The method in [16] exploit the Hawkes Point Process model with the Power-law kernel to model the self-exciting data and combines inferring the branching structure with optimizing model parameters by EM algorithm. In the E-step, they infer the branching structure and in the M-step, optimize parameters by the Newton-Raphson algorithm[6]. The framework of our method is similar to [16] but we use L-BFGS[10] for optimization in the M-step of the EM algorithm. Newton-Raphson algorithm can deal with convex optimization but will be unstable to optimize non-convex functions. By theoretical analysis and experiments, we find our target function in the M-step of EM algorithm is non-convex and Newton-Raphson algorithm cannot effectively deal with it.

2.2 Uncovering Hidden Social Graphs

For uncovering the diffusion structure, there exist numerous different methods which can be separated into two classes: 1) NETINF and its extension[3, 4, 14]; 2) the Hawkes Point Process and the mixture of Hawkes Point Processes[15, 1, 8, 9]. NETINF starts with independent points without edges and iteratively selects an edge which increase most significantly the log-likelihood of the network until the number of edges is equal to a target value. This method needs to calculate the probabilities of edges but select edges based on improvement of the log-likelihood so it is kind of different from distribution-based methods. However, this kind of methods needs lots of cascades occurring on a retweet network to recover it precisely[3]. The Hawkes Point Process model with the branching structure is actively used to uncover the diffusion structure, where hidden variables are introduced to represent the probabilities of the branching structure[15, 1, 8]. To calculate the hidden variables, people use EM algorithm to combine calculating hidden variables with optimizing model parameters. The mixture of Hawkes Point Processes model assumes multiple cascades influence each other and trys to model not only the branching structure in each cascade but also the influence between cascades [8]. Notably, the frameworks of the mixture of Hawkes Point Processes models are similar to the frameworks of the single Hawkes Point Processes model.

2.3 EM Variants

Apart from work on models, a great deal of work is on how to optimize the parameters of these models. With introducing the hidden variables to represent the branching structure and even the influence between cascades, the complexity of the target functions is increased so that most optimization methods fail to deal with them. In order to optimize model parameters more quickly and improve the performance of models, two EM- algorithm-based methods, Model Independent Stochastic Declustering (MISD) [11] and Maximum Penalized Likelihood Estimation (MPLE)[7], are designed with non-parametric estimations of the Hawkes Point Process in the M-step. MISD discretizes the variables of the intensity function and estimate the intensity function on discretized cells using the kernel density estimation. How to update the probabilities of parenthood relations is not changed in MISD. Notably, MISD is only effective for Hawkes Point Process with time-homogeneous background rate. Extending MISD, MPLE is similar to MISD but adds the penalty into the log-likelihood to improve the optimization and the change requires Euler-Lagrange Equations to solve the optimization problem[7]. MPLE is extended to optimize parameters of the mixture of Hawkes Point Processes in [17]. This project has a specific Hawkes Point Process model so we do not consider using these non-parametric methods for our optimization.

3 Our Proposed Method

This section elaborates the Hawkes Point Process model with the Power-law kernel and the branching structure of Hawkes Point Process model and we also explain how to use EM algorithm to combine inferring retweet networks and optimizing model parameters.

3.1 Hawkes Point Processes

Stochastic Point Process models are widely used to describe events occurring at random locations and/or times. Hawkes Point Process[5] is a special Stochastic Point Process model, where occurring points make future events more likely to occur. In our project, we assume each retweet occurs randomly and existing tweet will increase the possibility of occurrence of future retweets. A Hawkes Point Process model can be described by its intensity function:

$$\lambda(t) = \sum_{t_i<t} \phi_{m_i}(t - t_i) \tag{3}$$

where $\lambda(t)$ represents the probability of an event occurring at around time t given the historical events $\{(m_i, t_i)\}$, $t_i \leq t$.

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{P([t, t + \Delta t] | \{(t_i, m_i)\}, i = 1, ..., n)}{\Delta t} \tag{4}$$

The influence from the previous event (t_i, m_i) on the event (t, m) is quantified as $\phi_{m_i}(t - t_i)$ where we take the local user influence (i.e., the number of followers, m_i), influence decay and virality of the tweet content into account. The specific form of $\phi_{m_t}(t - t_i)$ is borrowed from [12]:

$$\phi_{m_i}(t - t_i) = \kappa m_i^{\beta}(t - t_i + c)^{-(1+\theta)} \tag{5}$$

where κ quantifies the virality of the tweet content, m_i^{β} quantifies local user influence in the exponential form and $(t - t_i + c)^{-(1+\theta)}$ represents the dacay of user influence with time. Note that κ, β, c and θ have a lower bound of zero. By now, we have had parameters of Hawkes Point Process models and p_{ij} describing the probabilities of parenthood relations. These parameters are summarized in Table 1.

Table 1: Model parameters and hidden variables

Quantity	Range	Meaning
κ	>0	virality of tweet contents. High value reflects the tweet is more likely to generate more retweets.
β	>0	the exponent in local user influence.
c	>0	prevent $\Phi_m(\tau)$ from being infinity when $\tau \approx 0$.
θ	>0	the power-law exponent.
p_{ij}	[0,1]	the probability of the tweet i being caused by the tweet j

3.2 Expectation Maximization Algorithm

EM algorithm is widely used to optimize the statistical models involving observed and hidden data, denoted as X and Z respectively. It repeats two steps, i.e., the E-step and the M-step and iteratively update model parameters denoted as Θ and hidden variables Z. In the E-step, the expectation of the complete log-likelihood with respect to the conditional distribution Z given X under the current estimate of model parameters:

$$Q(\Theta \mid \Theta^{old}) = E_{Z|X,\Theta^{old}}[logL(\Theta; X, Z)] \tag{6}$$

where $L(\Theta; X, Y)$ is the likelihood of the observed and latent data. In the M-step, EM algorithm optimizes the model parameters by maximizing $Q(\Theta \mid \Theta^{old})$.

$$\Theta^{new} = \underset{\theta}{\operatorname{argmax}}\, Q(\Theta \mid \Theta^{old}) \tag{7}$$

In order to infer the real retweet network and optimize the model parameters, we use EM algorithm to update p_{ij}, $i = 1 : n$, $j = 1 : i-1$ in the E-step and optimize κ, β,c, and θ in the M-step. Firstly, we derive $Q(\Theta \mid \Theta^{old})$ by using Equation 3 to 5. Given a cascade of retweets, $\{(m_i, t_i)\}$, $i = 1, ..., n$, the detailed derivation is shown:

$$\begin{aligned}
Q(\Theta \mid \Theta^{old}) &= E[logp(X, Y \mid \Theta) \mid X, \Theta^{old}] \\
&= \sum_{i=2}^{n}\sum_{j=1}^{i-1} p_{ij}[\log \Phi_{m_j}(t_i - t_j) - \int_{t_{i-1}}^{t_i} \lambda(t)dt] \\
&= \sum_{i=2}^{n}\sum_{j=1}^{i-1} p_{ij} \log \Phi_{m_j}(t_i - t_j) - \int_{0}^{T} \lambda(t)dt \\
&= \sum_{i=2}^{n}\sum_{j=1}^{i-1} p_{ij}[\log \kappa + \beta \log m_i - (1+\theta)\log(t_i - t_j + c)] + \\
&\sum_{i=1}^{n} \kappa m_i^{\beta} \frac{1}{\theta}[\frac{1}{(T - t_i + c)^{\theta}} - \frac{1}{c^{\theta}}]
\end{aligned} \tag{8}$$

where $p(X, Y \mid \Theta)$ is the complete likelihood with respect of Θ which can be derived based on the log-likelihood in [2](Ch.7.2). After getting $Q(\Theta \mid \Theta^{old})$, we optimize the model parameters by maximizing $Q(\Theta \mid \Theta^{old})$. In this project, we try two methods for maximization, i.e., Newton-Raphson algorithm[6] and L-BFGS[10]. Newton-Raphson algorithm use Laplacian approximation to get the approximated form of the local $Q(\Theta \mid \Theta_{old})$. By maximizing the derivative of the approximated $Q(\Theta \mid \Theta^{old})$, we can get the optimal Θ in respect of the Jacobian and Hessian matrices of $Q(\Theta \mid \Theta^{old})$, i.e.,

$$\Theta = \Theta^{old} - \mathbf{H}^{-1}\mathbf{J} \tag{9}$$

where $\mathbf{H}$ and $\mathbf{J}$ are the Hessian matrix and the Jacobian matrix of $Q(\Theta \mid \Theta^{old})$. The Hessian matrix and the Jacobian matrix are:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 Q(\Theta \mid \Theta^{old})}{\partial\kappa\partial\kappa} & \frac{\partial^2 Q(\Theta \mid \Theta^{old})}{\partial\kappa\partial\beta} & \frac{\partial^2 Q(\Theta \mid \Theta^{old})}{\partial\kappa\partial c} & \frac{\partial^2 Q(\Theta \mid \Theta^{old})}{\partial\kappa\partial\theta} \\ \frac{\partial^2 Q(\Theta \mid \Theta^{old})}{\partial\beta\partial\kappa} & \frac{\partial^2 Q(\Theta \mid \Theta^{old})}{\partial\beta\partial\beta} & \frac{\partial^2 Q(\Theta \mid \Theta^{old})}{\partial\beta\partial c} & \frac{\partial^2 Q(\Theta \mid \Theta^{old})}{\partial\beta\partial\theta} \\ \frac{\partial^2 Q(\Theta \mid \Theta^{old})}{\partial c\partial\kappa} & \frac{\partial^2 Q(\Theta \mid \Theta^{old})}{\partial c\partial\beta} & \frac{\partial^2 Q(\Theta \mid \Theta^{old})}{\partial c\partial c} & \frac{\partial^2 Q(\Theta \mid \Theta^{old})}{\partial c\partial\theta} \\ \frac{\partial^2 Q(\Theta \mid \Theta^{old})}{\partial\theta\partial\kappa} & \frac{\partial^2 Q(\Theta \mid \Theta^{old})}{\partial\theta\partial\beta} & \frac{\partial^2 Q(\Theta \mid \Theta^{old})}{\partial\theta\partial c} & \frac{\partial^2 Q(\Theta \mid \Theta^{old})}{\partial\theta\partial\theta} \end{bmatrix} \tag{10}$$

$$\mathbf{J} = \begin{bmatrix} \frac{\partial Q(\Theta \mid \Theta^{old})}{\partial\kappa} \\ \frac{\partial Q(\Theta \mid \Theta^{old})}{\partial\beta} \\ \frac{\partial Q(\Theta \mid \Theta^{old})}{\partial c} \\ \frac{\partial Q(\Theta \mid \Theta^{old})}{\partial\theta} \end{bmatrix} \tag{11}$$

The next step is the E-step where we use model parameters to update hidden variables. A hidden variable p_{ij} represents the probability of the i_th retweet being from the j_{th} retweet or tweet and $\Phi_{m_j}(t_i - t_j)$ is the probability of the j_{th} causing an event occurring at t_i so p_{ij} can be calculated as the fraction of $\Phi_{m_j}(t_i - t_j)$ over $\sum_{j=1}^{i-1} \Phi_{m_j}(t_i - t_j)$, i.e.,

$$\begin{aligned} p_{ij} &= \frac{\phi_{m_j}(t_i - t_j)}{\sum_{j=1}^{i-1} \phi_{m_j}(t_i - t_j)} \\ &= \frac{\kappa m_j^{\beta}(t_i - t_j + c)^{-(1+\theta)}}{\sum_{k=1}^{i-1} \kappa m_k^{\beta}(t_i - t_k + c)^{-(1+\theta)}} \end{aligned} \tag{12}$$

The M-step and E-step are repeated until $Q(\Theta \mid \Theta^{old})$ converges. The final κ, β, c and θ are optimal model parameters and we infer the retweet network according to final $\{p_{ij}\}$. For any i, p_{ij},$j = 1, ..., i-1$, the highest p_{ij} reflects the most possible parent of the tweet i.

Finally, we get a method based on EM algorithm, in which the M-step calculates the optimal model parameters κ, β, c, θ and the E-step the hidden variables $\{p_{ij}\}$. We show the pseudocode of our method in Table 1. By now, we have built completely our EM-algorithm-based method and we call our method H.EM and the pseudocode of H.EM is shown in Algorithm 1.

Algorithm 1 H.EM

```
1: procedure HEM({$t_i$},{$m_i$}, maxIteration)
2:     Initialize $\kappa$, $\beta$, c and $\theta$
3:     for ite=1:maxIteration do
4:         for i=1:n do
5:             for j=1:i-1 do
6:                 $p_{ij} = \frac{\phi_{m_j}(t_i - t_j)}{\sum_{j=1}^{i-1} \phi_{m_j}(t_i - t_j)}$
7:             end for
8:         end for
9:         $(\kappa, \beta, c, \theta) = argmax_{(\kappa,\beta,c,\theta)} Q(\Theta \mid \Theta^{old})$
10:        if $Q(\Theta \mid \Theta^{old})$ converges then
11:            break
12:        end if
13:    end for
14:    return $\kappa$,$\beta$,c,$\theta$ and {$p_{ij}$}
15: end procedure
```

4 Dataset

In this section, we introduce the constructed Twitter dataset that we use in our experiments. The work in this part consists of two components, collecting retweet cascades by the Twitter crawler and downloading friend lists of users in the cascades by the Twitter API.

4.1 Collecting Retweet Cascades by the Twitter Crawler

The Twitter crawler collects retweet cascades mentioning news articles from Sydney Morning Herald. Tweets collected in an hour are saved in a compressed file and we extract them to build complete cascades. The complete cascade is defined as a cascade whose tweets are not missing. An incomplete cascade containing tweets some of whose precedents are unknown and the absence of information of precedents makes it impossible to find the parenthood relations. To practically determine whether a cascade is complete, we need to know the format of tweets and what information is included in tweets. Tweets gotten from the Twitter crawler and the Twitter API have the same format[1] and they contain the owner of this tweet, the time when and the location where the tweet is released, the tweet content, whether this is a retweet and so on. `"retweeted_status/retweet_count"` of tweets can be used to determine the completeness of cascades because it reflects how many times the root tweet has been retweeted. The steps we take to find and construct complete cascades are: (1) find all tweets whose `"retweeted_status/retweet_count"` is 1 and use the root tweet's id as the name of the cascade to create a file to save it. Here, the information of the root tweet is contained in `"retweeted_status` and only tweets with `retweet_count"` equal to 1 are going to be saved. (2) For any tweet whose `"retweeted_status/retweet_count"` is greater than 1, we check whether the cascade it belongs to contains all its precedents. For example, the `"retweeted_status/retweet_count"` of a tweet is 10 and we find the file whose name is the root tweet's id to check whether tweets saved there have `"retweeted_status/retweet_count"` from 1 to 9. If true, we append it to that file. In this way, all tweets can be arranged as complete cascades.

4.2 Downloading Friend Lists by the Twitter API

Simultaneously, we extract all users of retweets in the complete cascades and download their friend lists by Twitter API[2]. Users of retweets can be found in `user_id` or `user_id_str`. The main challenge in construction of the Twitter dataset is the rate limitation from Twitter API, which allows to download friend lists of one user per minute. The possible way to deal with it is to use multiple Twitter accounts to download friend lists in parallel. So, I use two twitter accounts to speed up the process. Notably, it is better to download friend lists as soon as a tweet is saved because the friend lists are dynamic and future friend lists may be different from the current ones.

This system has been running since 14th February and by now, we have gotten around 68,000 cascades which contain totally around 260,000 tweets and 61,000 users (see details in Table 2). The number of cascades is too huge so we choose 274 cascades with more than 50 tweets among them as our experiment data. The number of users and tweets in chosen cascades is around 16,000 and 34,000 and almost all friend lists of these users have been downloaded.

[1]https://dev.twitter.com/overview/api/tweets
[2]https://dev.twitter.com/rest/public

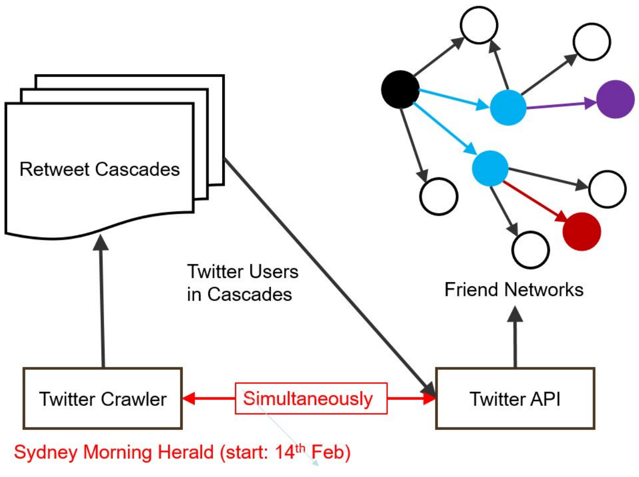

Figure 4:

Table 2: Statistics on downloaded data

Item	Quantity
Cascades	68040
Tweets in cascades	259186
Users in cascades	61174
Cascades with more than 50 Tweets (C_{50})	274
Users in C_{50}	16125
Tweets in C_{50}	33539
Downloaded friends of users in C_{50}	16051

5 Experiment and Evaluation

5.1 Experiments on Synthetic data: Comparisons with Maximum Likelihood Estimation

EM algorithm assumes the existence of hidden variables and results in a more complex target function: the expectation of the complete log-likelihood can be more difficult to maximize than the observed log-likelihood [12]. In order to analyze how well our method can optimize model parameters, we conduct experiments on the synthetic data and compare our method with Maximum Likelihood Estimation (MLE) used in [12]. Maximum Likelihood Estimation of a point process can be written as:

$$(\kappa, \beta, c, \theta) = \underset{(\kappa,\beta,c,\theta)}{\operatorname{argmax}} \sum_{i=2}^{n} log\lambda(t_i) - \int_{t_1} t_n \lambda(t) dt \tag{13}$$

the point process consists of $\{(t_i, m_i)\}$,$i = 1, ..., n$.

The synthetic data for experiments are generated by simulation of Hawkes Point Process. Totally 10 cascades are generated and each cascade is generated by simulation with different parameters. We conduct 20 experiments on each cascades with different initial parameters but H.EM and MLE always have the same initial parameters and iteration times. To maximize the expectation of the complete log-likelihood in the M-step of EM algorithm, L-BFGS and Newton-Raphson algorithm are tried respectively. However, Newton-Raphson algorithm is computationally unstable when dealing with the non-convex function[6]. The computational instability causes the determinant of Hessian matrix in Equation 9 close to zero and as a result, the Hessian matrix is non-invertible and Equation 9 cannot be computed. Thus, we did not get the optimal parameters from Newton-Raphson algorithm in the experiments.

Furthermore, we theoretically analyze the convexity of $Q(\Theta \mid \Theta^{old})$ by exploiting the first-order conditions of the convex function: a real-valued differentiable function f is convex **iff**

$$f(\mathbf{X}) \geq f(\mathbf{Y}) + \bigtriangledown f(\mathbf{Y})^T(\mathbf{Y} - \mathbf{X}) \tag{14}$$

for all $\mathbf{X}, \mathbf{Y} \in R^n$. $\bigtriangledown f(\mathbf{Y})$ is the Jacobian matrix of f. In our project, the function f is $Q(\Theta \mid \Theta^{old})$. To prove $Q(\Theta \mid \Theta^{old})$ is non-convex, we select $\mathbf{Y_0}$ making $f(\mathbf{Y_0})$ and $\bigtriangledown f(\mathbf{Y_0})$ finite:

$$f(\mathbf{Y_0}) \neq \infty$$

$$\bigtriangledown f(\mathbf{Y_0}) \neq \infty$$

According to the expression of $Q(\Theta \mid \Theta^{old})$, we know when $\kappa \to 0$, $Q(\Theta \mid \Theta^{old}) \to -\infty$.

$$\lim_{\kappa \to 0} \sum_{i=2}^{n} \sum_{j=1}^{i-1} p_{ij}[\log \kappa + \beta \log m_i - (1+\theta)\log(t_i - t_j + c)] + \sum_{i=1}^{n} \kappa m_i^{\beta} \frac{1}{\theta}[\frac{1}{(T - t_i + c)^{\theta}} - \frac{1}{c^{\theta}}] = -\infty$$

Thus, we select a $\mathbf{X_0}$ whose $\kappa \approx 0$ and we get

$$f(\mathbf{Y_0}) + \bigtriangledown f(\mathbf{Y_0})^T(\mathbf{Y_0} - \mathbf{X_0}) \neq \infty$$

$f(\mathbf{X_0})$ approximates negative infinity when $|\kappa|$ is small enough so we can find a $\mathbf{X_0}$ which makes

$$f(\mathbf{X_0}) \leq f(\mathbf{Y_0}) + \bigtriangledown f(\mathbf{Y_0})^T(\mathbf{Y_0} - \mathbf{X_0})$$

Thus, $Q(\Theta \mid \Theta_{old})$ is non-convex. In the similar way, we can prove it is non-concave . Thus, finally we prove $Q(\Theta \mid \Theta^{old})$ is non-convex and non-concave and optimization of $Q(\Theta \mid \Theta)$ is a non-convex problem. Although Newton-Raphson cannot optimize $Q(\Theta \mid \Theta^{old})$ effectively, L-BFGS makes it and get the optimal model parameters, which is also used in MLE to maximize the log-likelihood.

The steps to conduct experiments are explained here. Firstly, the model parameters are initialized by randomly sampling from the interval $[0.0001, 1]$ and H.EM and MLE share the same initial parameters in the each experiment. However, different experiments have different initial parameter. The iteration times of H.EM takes iterations of two steps in EM algorithm and of L-BFGS into account. The iteration times of E-step and M-step are 1000 and of L-BFGS are 10 iterations. Thus, the total iteration times in H.EM is 10,000 and to make results from H.EM and MLE comparable, we set iteration times in L-BFGS of MLE 10,000 as well. Finally, we get 20 pairs of optimal model parameters from experiments on each cascade (totally 200 pairs of model parameters are obtained).

Here, model parameters of a cascade are selected to display in Figure 5 to 8 where each boxplot contains the same parameter from a method. Parameters used in simulation (optimal parameters) are attached in the title of each image. Clearly shown in the figures, the parameters returned by both methods are very similar. For example, the difference between κ's from H.EM and MLE is at most 10^{-3} and for c, it is at most 10^{-4}. However, the variance of parameters from H.EM is slightly high. This is probably because adding latent variables into the likelihood makes the optimization problem more complex and L-BFGS in the EM algorithm cannot work as effective as that in MLE.

5.2 Seven Baselines

In order to evaluate the performance of H.EM in recovering retweet networks, we compare it with seven additional methods, i.e., 2 H.EM variants (1) Maximum Likelihood Estimation of Hawkes Point Processes with the Power-law kernel (H.MLEPL), (2) Maximum Likelihood Estimation of Hawkes Point Processes with the Exponential kernel (H.MLEEXP), 4 probability distributions: (3) Exponential distribution, (4) Power-law distribution, (5) Rayleigh distribution and (6) Social Exponential, and a probabilistic graphic model (7) NETINF [3]. Methods (1)-(6) infer the true parenthood relations by calculating p_{ij}, i.e., the probabilities of possible parenthood relations whereas NETINF gives the social graph based on the improvement of the likelihood of all cascades.

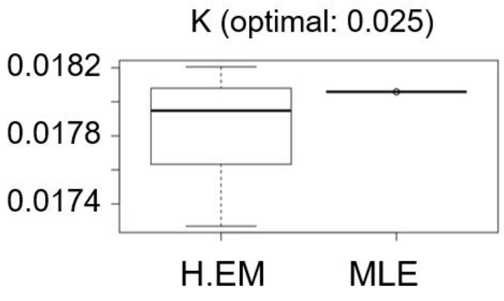

Figure 5: κ from H.EM and MLE

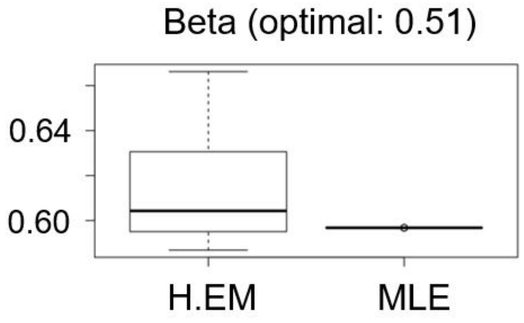

Figure 6: β from H.EM and MLE

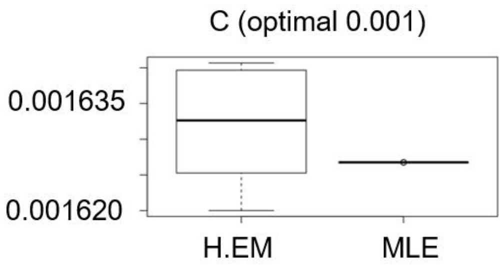

Figure 7: c from H.EM and MLE

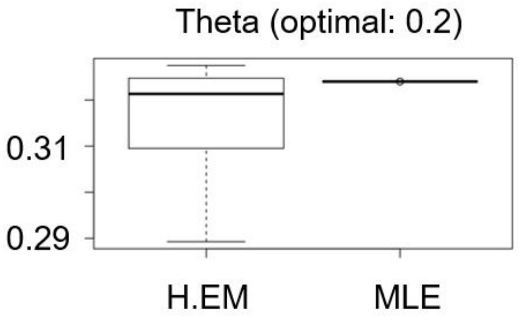

Figure 8: θ from H.EM and MLE

5.2.1 Two H.EM Variants

H.MLEPL is inspired by the results of experiments on the synthetic data: parameters from H.EM and MLE are close to each other. Instead of optimizing model parameters by EM algorithm, H.MLEPL gets optimal model parameters by MLE (i.e., by Equation 13) and then calculates $\{p_{ij}\}$ in the same way as that of H.EM:

$$p_{ij} = \frac{\kappa m_j^{\beta}(t_i - t_j + c)^{-(1+\theta)}}{\sum_{k=1}^{i-1} \kappa m_k^{\beta}(t_i - t_k + c)^{-(1+\theta)}}$$

Thus, there is no iteration in H.MLEPL. Similar to H.MLEPL, H.MLEEXP also optimizes model parameters by MLE and uses Equation 12 to compute p_{ij} but it uses the Exponential triggering kernel in the Hawkes Point Process model rather the Power-law kernel. The specific form of the Exponential triggering kernel is:

$$\phi_{m_i}(t - t_i) = e^{-\alpha(t-t_i)} \tag{15}$$

where α is the model parameter and the equation for calculating $\{p_{ij}\}$ is:

$$p_{ij} = \frac{e^{-\alpha(t-t_i)}}{\sum_{k=1}^{i-1} e^{-\alpha(t-t_k)}}$$

No iteration in H.MLEPL and H.MLEEXP makes them simpler and more efficient than H.EM.

5.2.2 Four p_{ij} Distribution Estimations

The four probability distributions are designed based on widely used distributions: the Power-law distribution, the Exponential distribution and the Rayleigh distribution. We describe the four probability distributions here:

(1) Exponential distribution (the subscript represents the tweet i being caused by the tweet j; the same hereinafter)

$$p_{i,j} = \alpha e^{-\alpha(t_i - t_j)}$$

(2) Power-law distribution

$$p_{i,j} = (\alpha - 1)(t_i - t_j)^{-\alpha}$$

(3) Rayleigh distribution

$$p_{i,j} = \alpha(t_i - t_j)e^{-0.5\alpha(t_i - t_j)^2}$$

(4) Social Exponential

$$p_{ij} = m_j e^{-\alpha(t_i - t_j)}$$

where α is the distribution parameter. The first three distributions only consider the distribution of inter-arrival times while the forth one combines the user influence with inter-arrival times. These four distributions can be used to directly calculate the probabilities p_{ij} without any iteration and for any i, the highest probability among $\{p_{i,j}\}$,$j = 1 : i - 1$, implies the most possible parent of the tweet i. We are going to select the optimal parameters for these four distributions in the calibration stage. Note that Social Exponential needs normalizing.

5.2.3 NETINF

NETINF also uses the probability distribution to calculate the possibility of edges in the retweet networks but it selects edges based on how much improvement an edge can bring about on the likelihood of retweet network. This method starts with independent points without edges and it select a link between two points as an edge of the retweet network, adding which into the retweet network can improve the likelihood of the retweet network most significantly. Then, from the left links not in the retweet network, NETINF continues selecting links causing the largest improvement in the likelihood and adding them into the retweet network until the number of edges in the retweet network attains a target value. NETINF has parameters in its probability distribution of the links so NETINF needs training to choose optimal distribution parameters. Notably, there are iterations in NETINF and an edge is selected in each iteration.

5.3 Performance Measures

We use two measures to evaluate the performance of our method, i.e., AUC and the accuracy. It is impossible to retrieve the real retweet networks so we use the friend networks as the ground truth to evaluate performance. H.EM, H.MLEPL, H.MLEEXP and four probability distributions can give a sequence of probabilities for each retweet to represent the possibilities of it being caused by its precedents. Based on the friend network, if there is an edge between two users, we give the true labels to probabilities of links between tweets of both users; otherwise, we give false labels. A Receiver Operating Characteristic (ROC) Curve can be drawn based on each sequence of labeled probabilities, the area under which is AUC. In this way, AUC of each retweet can be computed and we collect AUC's of all retweets in all cascades and calculate the mean AUC to evaluate the performance of different models. The larger mean AUC normally reflects the better performance.

However, NETINF does not infer edges of retweet networks based on probabilities so we cannot use AUC to evaluate its performance. Instead, we use the prediction accuracy as the measure, i.e., the fraction of correctly predicted links among all predicted links. In case of 2 H.EM variants and four p_{ij} distribution estimations, we predict the link with highest p_{ij} as the true edge in the retweet network. Like labeling the probabilities, here if the owners of tweets are friends then predicted links between these tweets are regarded to be correct.

5.4 Calibration of Parameters of Four p_{ij} Distribution Estimations and of NETINF

H.EM, H.MLEPL and H.MLEEXP do not need training and given a cascade, they maximize the log-likelihood to obtain optimal parameters. However, the distribution parameters in four probability-distribution-based methods and NETINF need choosing and these parameters play an important role in predicting the edges of the retweet networks. In this section, we optimize the distribution parameters of four probability distributions and NETINF by linear search. The chosen parameters should make four probability-distribution-based methods achieve the highest mean AUC's on the training dataset and NETINF achieve the largest accuracy.

20 cascades of retweets are randomly sampled from the whole dataset (274 complete cascades) as the calibration set and the iteration times in NETINF is set to one less than the number of tweets in the cascade because the number of links returned by NETINF is equal to its iteration times. The resulting parameters of Exponential distribution, Power-law distribution, Rayleigh distribution, Social Exponential and NETINF are 10^{-5}, 1.0001, 10^{-15}, 6.2×10^{-6} and 2.78 respectively.

5.5 Performance of H.EM, Two H.EM Variants, Four p_{ij} Distribution Estimations and NETINF

The remaining 254 retweet cascades are used as test dataset. Like the setting in the experiments on synthetic data, L-BFGS in the M-step of H.EM has 10 iterations and iteration times between the E-step and the M-step are 1000. Besides, we initialize the parameters of H.EM by sampling randomly in the interval $[0.0001, 1]$ and we set iteration times of NETINF as what we do in the training stage. Similarly, L-BFGS in H.MLEPL and H.MLEEXP have 10,000 iterations with random initial parameters to optimize model parameters. We keep using the accuracy and AUC to evaluate the performances, which are calculated in the same way as that in the training stage. Note that AUC is only applied to evaluating H.EM, two H.EM variants and four p_{ij} distribution estimations while the accuracy is applied on all. Finally, we get the test results shown in Figure 9 and 10 and each boxplot contains all AUC's or accuracies from an approach. We also calculate the mean AUC and accuracy to evaluate the performance of each approach, shown in Table 3.

In Table 3, Social Exponential performs best with the highest mean AUC of 0.872 and the second highest mean accuracy of 0.556. The parameter of Social Exponential is 6.2×10^{-6} which is a minuscule number so that the exponential distribution in Social Exponential is almost equal to 1. As a result, the influence of time is ignored in Social Exponential and it mainly depends on the local user influence. We theorize that this effect is due to our constructed dataset in which the first tweet is usually emitted by the official Twitter handle of Sydney Morning Herald and retweeted directly by most other users. We are currently gathering a more general-purpose dataset of diffusions to test this conclusion.

Our method is ranked the second with the second highest mean AUC of 0.832 and the third highest mean accuracy of 0.506. Although it does not perform best in the experiments, it recovers the parenthood relations without requiring calibrations. H.MLEPL has the similar performance with a close mean AUC of 0.83 and the mean accuracy of 0.468. This is because H.MLEPL uses MLE to optimize model parameters which has the similar ability to get optimal parameters to that of H.EM. However, the consistently better performances of H.EM means that inferring p_{ij} during the optimization of model parameters can more precisely infer the parenthood relations.

H.MLEEXP has the same exponential kernel as that of Social Exponential but H.MLEEXP performs much worse than Social Exponential. The weakness of the exponential distribution is also demonstrated by the small parameter of Social Exponential which causes the exponential distribution of Social Exponential to be almost ignored. Besides, Exponential distribution, Power-law distribution and H.MLEEXP have the similar performances, which can be explained by their models where the closer occurring time brings about the higher probability. This contrasts the characteristic of our dataset: users usually retweets the first tweet.

To our surprise, Rayleigh distribution gets the best performance in the measure of the accuracy with the mean accuracy of 0.566 but performs mediocrely under the measure of AUC. It has the parameter of 10^{-15} so the longer inter-arrival time gets the higher probability from the Rayleigh distribution. This kind of fits the characteristic of our dataset. Besides, NETINF does not have outstanding performance in the experiments, which is because it needs lots of cascades to recover the retweet networks.

Through the above analysis, we summarize our conclusions: (1) inferring p_{ij} during optimization of model parameters leads to the better performance; (2) our method is effective to recover retweet networks and it is possible to retrieve the parenthood relations without a calibration set of diffusions; (3) retweets in our dataset are usually retweeted from the first tweet.

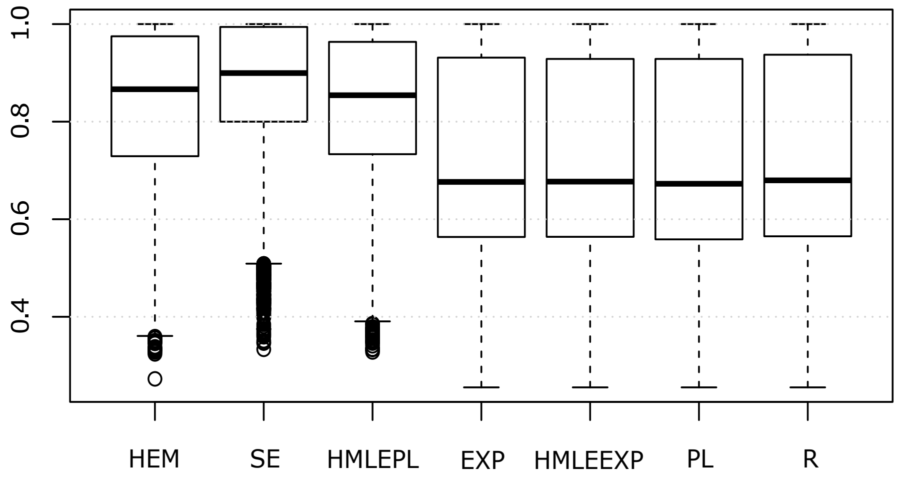

Figure 9: AUC's of H.EM and seven baselines on real data

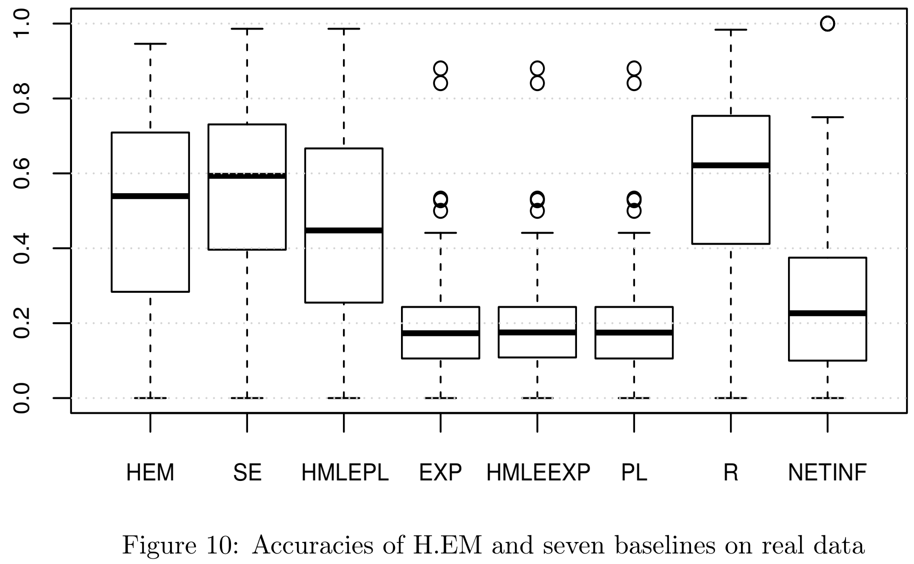

Figure 10: Accuracies of H.EM and seven baselines on real data

Table 3: Mean AUC and Accuracy Different Methods on Test Stage

	H.EM	SE	H.MLEPL	EXP	H.MLEEXP	PL	R	NETINF
Mean AUC	0.832	**0.872**	0.830	0.726	0.726	0.721	0.728	
Mean Accuracy	0.506	0.556	0.468	0.185	0.187	0.186	**0.566**	0.249

6 Conclusion

This project uses Hawkes Point Process model with the Power-law triggering kernel to model the self-exciting features of retweet cascades and exploits the branching structure to retrieve the true parenthood relations in the tweets. Optimizing model parameters and inferring retweet networks are combined by EM algorithm. The experiment dataset including the retweet cascades and the friend lists of users in cascades are all constructed ourselves. In the experiments on the synthetic data, we compare the parameters from both H.EM and MLE and find both methods optimize parameters similarly well, which is also demonstrate by the similar performance of both methods on the real data. In the experiments on the real data, we use two measures: AUC and the accuracy, and compare our method with seven baselines, i.e., two H.EM variants, four p_{ij} distribution estimations and NETINF. Through the comparison between H.EM and H.MLEPL, we find inferring p_{ij} during optimization of model parameters brings about better performance. Although our method does not outperform Social Exponential under both measures, it gets close performance without requiring a calibration set of diffusions so it can be used when there is no more cascade for training or for improving prediction. Besides, we also find a characteristic of our dataset: retweetes are usually retweeted from the first tweet. To improve our proposed method, we plan to try more competitive triggering kernels in the future work, which may be more effective than the Power-law kernels. We will also construct a more general-purpose dataset with diffusions to test our method as the future work.

7 Acknowledgment

This project is completed with help from Marian-Andrei Rizoiu, Swapnil Mishra and Yifei Zhang and I would like to express my gratitude to them. Marian-Andrei Rizoiu, as my supervisor, helps me to overcome obstacles I face in my project and corrects my wrong understanding. He also gives detailed comments to my presentation and report which help me to refine my presentation and report. Swapnil Mishra provides the Twitter crawler and some test dataset to my project. He also enthusiastically answers my questions on theories used in the project and points out the problems in my presentation. Social Exponential is the work of Yifei Zhang and its good performance makes it the future target of our project.

Besides, I would like to thank the course conveners, Weifa Liang and Peter Strazdins. In the weekly meeting, I learn how to make academic presentation and write academic report. They also give helpful advice on my final presentation.

Finally, I would like to thank the computational media group at ANU. They provide me the access to the high-performance server which accelerate my experiments and give advice on my research.

References

[1] Yoon-Sik Cho, Aram Galstyan, P Jeffrey Brantingham, and George Tita. Latent self-exciting point process model for spatial-temporal networks. *Discrete and Continues Dynamic Systems Series B*, 2013.

[2] Daryl J Daley and David Vere-Jones. *An introduction to the theory of point processes: volume I*. Springer Science & Business Media, 2007.

[3] Manuel Gomez Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1019–1028. ACM, 2010.

[4] Manuel Gomez Rodriguez, Jure Leskovec, and Bernhard Schölkopf. Structure and dynamics of information pathways in online media. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 23–32. ACM, 2013.

[5] Alan G Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, pages 83–90, 1971.

[6] Kenneth Lange. A gradient algorithm locally equivalent to the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 425–437, 1995.

[7] Erik Lewis and George Mohler. A nonparametric em algorithm for multiscale hawkes processes. *Journal of Nonparametric Statistics*, 2011.

[8] Liangda Li and Hongyuan Zha. Dyadic event attribution in social networks with mixtures of hawkes processes. In *Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management*, pages 1667–1672. ACM, 2013.

[9] Scott Linderman and Ryan Adams. Discovering latent network structure in point process data. In *International Conference on Machine Learning*, pages 1413–1421, 2014.

[10] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[11] David Marsan and Olivier Lengline. Extending earthquakes' reach through cascading. *Science*, 319(5866):1076–1079, 2008.

[12] Swapnil Mishra, Marian-Andrei Rizoiu, and Lexing Xie. Feature driven and point process approaches for popularity prediction. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1069–1078. ACM, 2016.

[13] Marian-Andrei Rizoiu, Lexing Xie, Scott Sanner, Manuel Cebrian, Honglin Yu, and Pascal Van Henteryck. Expecting to be hip: Hawkes intensity processes for social media popularity.

[14] Manuel Gomez Rodriguez and Bernhard Schölkopf. Submodular inference of diffusion networks from multiple trees. 2012.

[15] Aleksandr Simma and Michael I Jordan. Modeling events with cascades of poisson processes. *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence (UAI2010)*, 2010.

[16] Sharareh Taghipour and Dragan Banjevic. Trend analysis of the power law process using expectation–maximization algorithm for data censored by inspection intervals. *Reliability Engineering & System Safety*, 96(10):1340–1348, 2011.

[17] Ke Zhou, Hongyuan Zha, and Le Song. Learning triggering kernels for multi-dimensional hawkes processes. In *ICML (3)*, pages 1301–1309, 2013.