A RECURRENT NEURAL CASCADE-BASED MODEL FOR CONTINUOUS-TIME DIFFUSION PROCESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Many works have been proposed in the literature to capture the dynamics of diffusion in networks. While some of them define graphical markovian models to extract temporal relationships between node infections in networks, others consider diffusion episodes as sequences of infections via recurrent neural models. In this paper we propose a model at the crossroads of these two extremes, which embeds the history of diffusion in infected nodes as hidden continuous states. Depending on the trajectory followed by the content before reaching a given node, the distribution of influence probabilities may vary. However, content trajectories are usually hidden in the data, which induces challenging learning problems. We propose a topological recurrent neural model which exhibits good experimental performances for diffusion modelling and prediction.

1 INTRODUCTION

The recent development of online social networks enabled researchers to suggest methods to explain and predict observations of diffusion across networks. Classical cascade models, which are at the heart of the research literature on information diffusion, regard the phenomenon of diffusion as an iterative process in which information transits from users to others in the network (Saito et al., 2008; Gomez-Rodriguez et al., 2011), by a so-called word-of-mouth phenomenon. In this setting, diffusion modeling corresponds to learning probability distributions of content transmission. Various cascade models have been proposed in the literature, each inducing its own learning process to explain some observed diffusion episodes and attempting to extract the main dynamics of the network. However, most of these models rely on a strong markovian assumption for which the probabilities of next infections[1] only depend on whom is already infected at the current step, not on the past trajectories of the diffused content. We claim that the history of spread contains much valuable information that should be taken into account by the models. Past trajectories of the diffusion can give insights about the different natures of the contents. Also, the content may be changed during the diffusion, with different transformations depending on which users re-transmit the information.

On the other hand some recent approaches rely on representation learning and recurrent neural networks (RNN) to predict the future spread of diffusion given the past. A naive possibility would be to consider diffusion episodes as sequences of infections and propose temporal point process approaches to model the dynamics. Using the Recurrent Marked Temporal Point Process model (Du et al., 2016), the current hidden state of the RNN would embed the history of the whole diffusion sequence, which would be used to output the next infected node and its time of infection. However, since diffusion episodes are not sequences but trees, naive recurrent methods usually fail in capturing the true dynamics of the networks. Embedding the whole past in the state of a given node rather than restraining it to its specific ancestor branch leads to consider much independent and noisy events for future predictions. A model that would consider the true diffusion paths would be more effective to focus on the useful past. If the true diffusion paths were known, it would be possible to adapt works on recurrent neural models for tree structures such as successfully proposed in (Tai et al., 2015) for NLP tasks. Unfortunately, in most of applications the topology of diffusion is unknown while learning. The only observations available are the timestamps of the infected nodes.

[1]Throughout this paper, we refer to infection for denoting the participation of a node of the network in the diffusion.

To cope with it, (Wang et al., 2017a) proposed `Topo-LSTM`, a Long-Short Time Memory network that considers a known graph of relationships between nodes to compute hidden states and cells of infected nodes. The hidden state and cell of a given node at t depend on those from each of its predecessors that have been infected before t. Since nodes may have multiple predecessors that are infected at time t, the classical `LSTM` cannot be applied directly. Instead, (Wang et al., 2017a) proposed a cell function that aggregates infector candidate states via mean pooling. This allows to take the topology of the possible diffusion into account, but not the past trajectory of the content (it averages all possible paths). To overcome this, (Wang et al., 2017b) proposed a cascade attention-based RNN, which rather defined an neural attention mechanism to assign weights to infector candidates before summing their contribution in the new hidden state at t. The attention network is supposed to learn to identify from whom comes the next infection based on past states. However, such an approach is likely to converge to most of attention weight vectors in the center of the simplex, since diffusion is a stochastic process with mostly very weak influence probabilities. The deterministic inference process of the approach limits its ability to produce relevant states by mixing from multiple candidates rather than sampling the past trajectories from their posterior probabilities. Note the similar approach in (Wang et al., 2018), which does not uses RNN but defines a composition module of the past via attention networks to embed the episode in a representation space from which the probability of the next infected node can be deduced. Beyond the limits discussed above w.r.t. deterministic mixing of diffusion trajectories, no delay of infection is considered in this work, which makes it impossible to use for diffusion modeling purposes.

Recently, many works in representation learning used random walks on graphs to sample trajectories that can be used to learn a network embedding which respects some topological constraints. While `DeepWalk` (Perozzi et al., 2014) only uses structural information, models proposed in (Nguyen et al., 2018) or (Shi et al., 2018) include temporal constraints in random walks to sample feasible trajectories w.r.t. observed diffusion episodes. The approach `DeepCas` from (Li et al., 2016) applies this idea for the prediction of diffusion cascades. However in this paper we assume that no graph of relationships is available beforehand (or it is not representative of the true diffusion channels of the network ()). Moreover, no inference process is introduced in `DeepCas` to sample trajectories from their posterior probabilities given the observed diffusion sequences. The sampling of trajectories is performed in an initialization step, before learning the parameters of the diffusion model.

In this paper, we propose the first bayesian topological recurrent neural network for sequences with tree dependencies, which we apply for diffusion cascades modelling. Rather than building on a preliminary random walk process, the idea is to consider trajectory inference during learning to converge to better representations of the infected nodes. Following the stochastic nature of diffusion, the model infers trajectories distributions from observations of infections, which are in turn used for the inference of infection probabilities in an iterative learning process. Our probabilistic model, based on the famous continuous-time independent cascade model (`CTIC`) (Saito et al., 2009) is able to extract full paths of diffusion from sequential observations of infections via black-box inference, which has multiple applications in the field. Our experiments validate the potential of the approach for modeling and prediction purposes.

The remaining of the paper is structured as follows. Section 2 presents some background and notations of the approach. Section 3 presents the proposed model. Section 4 reports experimental results of the approach compared to various baselines.

2 BACKGROUND

2.1 INFORMATION DIFFUSION

Information diffusion is observed as a set of diffusion episodes $\mathcal{D}$. Classically, episodes considered in this paper only contain the first infection event of each node (the earlier time a content reached the node). Let $\mathcal{U} = \{u_0, u_1, ..., u_{N-1}\}$ be a set of N nodes, u_0 standing for the world node, used to model influences from external factors or spontaneous infections (as done in () for instance). A diffusion episode $D = (U^D, T^D)$ reports the diffusion of a given content in the network as a sequence of infected nodes $U^D = \left(U_0^D, ..., U_{|D|-1}^D\right)$ and a set $T^D = \{t_u^D \in \mathbb{N} + \infty | u \in \mathcal{U}\}$ of infection time-stamps for all $u \in \mathcal{U}$. U^D is ordered chronologically w.r.t. the infection time-stamps T^D. Thus, $U_i^D \in \mathcal{U}$ corresponds to the i-th infected node in D for all $i \in \{0, ..., |D| - 1\}$, with

$|D|$ the number of infected nodes in the diffusion. Every episode in $\mathcal{D}$ starts by the world node u_0 (i.e., $U_0^D = u_0$ for all episodes D). We note t_u^D the infection time-stamp in D for any node in $\mathcal{U}$, ∞ for nodes not infected in D. Time-stamps are relative w.r.t. to t_1^D, arbitrarily set to $t_1^D = 1$ in the data. Note that we also set $t_0^D = 0$ for every episode D. We thus have for every episode D: $t_{U_i^D}^D \leq t_{U_j^D}^D$ for all $(i, j) \in \{0, ..., |D| - 1\}^2$ such that $j > i$. In the following, $D_i = (U_i^D, t_{U_i^D}^D)$ for $iin\{0, ..., |D| - 1\}$ denotes the $i - th$ infected node in U with its associated time-stamp.

Cascades are richer structures than diffusion episodes, since they describe how a given diffusion happened. Note that, while several transmission events to a same given destination node might succeed during the diffusion process, the cascade structure only stores the first transmission event $(u \to v)$ that succeeded from any node u to the destination node v. A cascade $C^D = (U^D, T^D, I^D)$ thus corresponds to a transmission tree rooted in u_0 and reaching nodes in U^D during the diffusion, according to a sequence I^D of infector indices in U^D: for any $j \in \{1, ..., |D| - 1\}$ and any $i \in \{0, ..., |D|-2\}$, I_j^D equals i iff U_j^D was infected by U_i^D in the diffusion D (i.e., U_i^D is the infector of U_j^D). We arbitrarily set $I_0^D = 0$ (no infector for the world node). Note that I_1^D is always equal to 0, since there is no other candidate for being the infector of U_1^D than the world u_0. We have: $U_j^D < j$ for every D and every $j \in \{1, ..., |D| - 1\}$. For convenience, we note $I^D(i) \in U^D$ the ancestor of U_i^D for $i \in \{0, ..., |D| - 1$. Also, note that the cascade structures respect that $t_{I^D(j)}^D < t_{U_j^D}^D$ for every D and all $j \in \{1, ..., |D| - 1\}$ (the infector of a node v is mandatory a node that was infected before v).

2.2 Cascade Models

The Independent Cascade model (IC) (Goldenberg et al., 2001) considers the spread of diffusion as cascades of infections over the network. We focus in this paper on cascade models such as IC, which tend to reproduce realistic temporal diffusion dynamics on social networks (Guille et al., 2013). The classical version of IC is an iterative process in which, at each iteration, every newly infected user u gets a unique chance to infect other users v of the network with some probability $\theta_{u,v}$. The process iterates while new infections occur. Since the expectation-maximization algorithm proposed in Saito et al. (2008) to learn its parameters, IC has been at the heart of diffusion works.

However, in real life, diffusion indeed occur in continuous time, not discrete as assumed in IC. (Lamprier et al., 2016) proposed `DAIC`, a delay-agnostic version of `IC`, where diffusion between nodes is assumed to follow uniform delay distributions rather than occurring in successive discrete time-steps. A representation learning version of `DAIC` has been proposed in (Bourigault et al., 2016), where nodes are projected in a continuous space in a way that the distance between node representations render the probability that diffusion can occur between them. This allowed the authors to obtain a more compact and robust model than the former version of `DAIC`. Beyond uniform time delay distributions, two main works deal with continuous-time diffusion. `NetRate` (Gomez-Rodriguez et al., 2011) learns parametric time-dependent distributions to best fit with observed infection time-stamps. As `NetRate`, `CTIC` (Saito et al., 2009) uses exponential distributions to model delays of diffusion between nodes, but rather than a single parameter for each possible relationship, delays and influence factors are considered as separated parameters, which leads to more freedom w.r.t. to observed diffusion tendencies. Delays and influence parameters are learned conjointly by an EM-like algorithm. Note the continuous-time cascade model extension in (Zhang et al., 2018), which embeds users in a representation space so that their relative positions both respect some structural community properties and can be used to explain infection time-stamps of users.

In our model we consider that the state of a node v in an episode D depends on the state of the node u who first transmitted the content to v. Therefore, we need to rely on a continuous-time model such as `CTIC` (Saito et al., 2009), which serves as a basis for our work.

3 Recurrent Neural Diffusion Model

This section first presents the recurrent generative model of cascades. Then, it details the proposed learning algorithm.

3.1 RECURRENT DIFFUSION MODEL

As discussed above, we consider that each infected node v in an episode D owns a state $\phi_v^D \in \mathbb{R}^d$ depending on the path the content followed to reach v in D, with d the dimension of the representation space. Knowing the state ϕ_u^D of the node u that first infected v, the state ϕ_v^D is computed as:

$$\phi_v^D = \phi(\phi_u^D, \omega_v^{(\phi)})$$

with $\phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ a function that transforms the state of u according to a shared representation $\omega_v^{(\phi)} \in \mathbb{R}^d$ of the node v. This function can either be an Elman RNN cell, multi-layer perceptron (MLP) or a Gated Recurrent Unit (GRU). An LSTM could also be used here, with ϕ_v^D containing both the cell and the state of v in D. We use the Elman RNN cell in our experiments:

$$\phi_v^D = \tanh(\phi_u^D \theta_h^{(\phi)} + b_h^{(\phi)} + \omega_v^{(\phi)} \theta_i^{(\phi)} + b_i^{(\phi)}) \tag{1}$$

where $\theta_h^{(\phi)}$ and $\theta_i^{(\phi)}$ are two $(d \times d)$ parameter matrices and $b_h^{(\phi)}$ and $b_i^{(\phi)}$ are two bias vectors of size d to be learned.

Given a state ϕ for an infected node u in D, the probability that u infects v in the future is given by:

$$k_{u,v}(\phi) = \sigma\left(\left(\gamma^{(k)}\phi + (1-\gamma^{(k)})\omega_u^{(k,1)}\right)\omega_v^{(k,2)}\right) \tag{2}$$

where $\sigma(.)$ stands for the sigmoid function, $\omega_u^{(k,1)}$ and $\omega_u^{(k,2)}$ correspond to two embeddings of size d for any node u, and $\gamma^{(k)} \in [0; 1]$ is a scalar weight that allows us to set the importance of the past in the probability computations. A weight $\gamma^{(k)} = 0$ leads to ignore the past and we get a simple embedded version of CTIC. In that case, $k_{u,v}(\phi)$ comes down to the sigmoid of the dot product between an embedding $\omega_u^{(k,1)}$ of the emitter u and an embedding $\omega_v^{(k,2)}$ of the receiver v. A weight $\gamma^{(k)} = 1$ gives everything to the history of the ancestor branch of the node u. In our experiments, $\gamma^{(k)}$ is set automatically by taking $\gamma^{(k)} = \sigma(\omega^\gamma)$, where ω^γ corresponds to a scalar parameter to be learned. This allows the learning process to be guided in the early steps, when the history states are too unstable (in all our experiments, ω^γ falls in the first epochs, to finally grow to high values when some stability has been reached.

Similarly to the CTIC model, if a node u succeeds in infecting another node v, the delay of infection depends on an exponential distribution with parameter $r_{u,v}$. To simplify learning, we assume that the delay of infection does not depend on the history of diffusion, only the probability of infection does. Thus, for a given pair (u, v), the delay parameter is the same for every episode D:

$$r_{u,v} = exp(-|\omega_u^{(r)}\omega_v^{(r)}|) \tag{3}$$

with $|x|$ the absolute value of a real scalar x and $\omega_u^{(r)}$ corresponds to an embedding of size d for any node $u \in \mathcal{U}$.

We set $\Theta = \left(\phi_0, \omega^\gamma, (\omega_u^{(\phi)})_{u\in\mathcal{U}}, (\omega_u^{(k,1)})_{u\in\mathcal{U}}, (\omega_u^{(k,2)})_{u\in\mathcal{U}}, (\omega_u^{(r)})_{u\in\mathcal{U}}, \theta_h^{(\phi)}, \theta_i^{(\phi)}, b_h^{(\phi)}, b_i^{(\phi)}\right)$ as the parameters of our model. The generative process, similar to the one of CTIC, is given in appendix. In this process, the state of the initial node, the world node u_0, is a parameter vector ϕ_0 to be learned. The process iterates while there remains some nodes in a set of infectious nodes (initialized with u_0). At each iteration, the process selects the infectious node u with minimal time-stamp of infection, removes it from the set of infectious nodes, records it as infected and attempts to infect each non infected node v according to the probability $k_{u,v}(\phi_u^D)$. If it succeeds, a time t is sampled for v with an exponential law with parameter $r_{u,v}$. If the new time t for v is lower than its current time t_v^D (initialized with $t_v^D = \infty$), this new time is stored in t_v^D, v is added to the set of infectious nodes and its new state ϕ_v^D is computed according to its new infector u.

3.2 LEARNING THE MODEL

As in CTIC, we need to define the probability that the node $u \in U^D$ infects the node $v \in U^D$ at time t_v^D with our model. Given a state ϕ for u in D, we have:

$$a_{u,v}^D(\phi) = k_{u,v}(\phi) r_{u,v} \exp^{-r_{u,v}(t_v^D - t_u^D)} \tag{4}$$

Also, the probability that u does not infect v before t_v^D given a state ϕ for u in D is:

$$b_{u,v}^D(\phi) = 1 - k_{u,v}(\phi) \int_{t_u^D}^{t_v^D} r_{u,v} \exp^{-r_{u,v}(t-t_u^D)} dt = k_{u,v}(\phi) \exp^{-r_{u,v}(t_v^D - t_u^D)} + 1 - k_{u,v}(\phi) \quad (5)$$

The probability density that node v is infected at time t_v^D given a set of states $\boldsymbol{\phi}$ for all nodes infected before v is:

$$h_v^D(\boldsymbol{\phi}) = \sum_{u \in \mathcal{U}, t_u^D < t_v^D} a_{u,v}^D(\phi_u) \prod_{x \in \mathcal{U} \setminus \{u\}, t_x^D < t_v^D} b_{x,v}^D(\phi_x) \quad (6)$$

$$= \prod_{x \in \mathcal{U}, t_x^D < t_v^D} b_{u,v}^D(\phi_x) \sum_{u \in \mathcal{U}, t_u^D < t_v^D} a_{u,v}^D(\phi_u) / b_{u,v}^D(\phi_u) \quad (7)$$

where ϕ_u is the state of node u in $\boldsymbol{\phi}$. Similarly, the probability density that node v is not infected in D at the end of observation time T given a set of states $\boldsymbol{\phi}$ for all nodes infected in D is:

$$g_v^D(\boldsymbol{\phi}) = \prod_{u \in U^D} (k_{u,v}(\phi_u) \exp^{-r_{u,v}(T - t_u^D)} + 1 - k_{u,v}(\phi_u)) \approx \prod_{u \in U^D} (1 - k_{u,v}(\phi_u)) \quad (8)$$

where the approximation is done assuming a sufficiently long observation period.

The learning process of our model is based on likelihood maximization. However, since observations only contain infection time-stamps, this requires to marginalize over every possible sequence of ancestors I for every $D \in \mathcal{D}$ to be able to compute the states of nodes used in our model:

$$\log p(\mathcal{D}) = \sum_{D \in \mathcal{D}} \log p(D) = \sum_{D \in \mathcal{D}} \log \sum_{I \in \mathcal{I}^D} p(D, I) \quad (9)$$

where $\mathcal{I}^D$ is the set $\{v \in \mathbb{N}^{|D|-1} | v_0 = 0 \wedge \forall i > 0, v_i < i\}$ of all possible ancestors sequences for D. $p(D, I)$ corresponds to the joint probability of the episode D and an ancestor sequence $I \in \mathcal{I}^D$. Taking $p(D, I) = p(I)p(D|I)$ would lead to an intractable computation of $p(D|I)$ using our recurrent cascade model, since it would imply to estimate the probability of any infection in D according to the full ancestors sequence. Hopefully, using the bayesian chain rule, the joint probability can be given by:

$$p(D, I) = \prod_{i=1}^{|D|-1} p(D_i | D_{<i}, I_{<i})\, p(I_i | D_{\leq i}, I_{<i}) \prod_{v \notin U^D} p(v \notin U^D | D_{\leq |D|-1}, I) \quad (10)$$

where $D_{<i} = (D_j)_{j \in \{0,...,i-1\}}$ corresponds to the sequence of the i first components of D (the i first components in U^D with their associated time-stamps) and $I_{<i} = (I_j)_{j \in \{0,...,i-1\}}$ stands for the vector containing the i first components of I. We have for every $i \in \{1, ..., |D| - 1\}$: $p(D_i | D_{<i}, I_{<i}) = h_{U_i^D}^D(\boldsymbol{\phi}_{<i}^D)$, where $\boldsymbol{\phi}_{<i}^D$ is a set containing the states of the i first infected nodes in D, which can be deduced from $D_{<i}$ and $I_{<i}$ using the equation 1. We also have: $p(v \notin U^D | D_{\leq |D|-1}, I) = g_v^D(\boldsymbol{\phi}_{\leq i}^D)$. The probability $p(I_i | D_{\leq i}, I_{<i})$ is the conditional probability that $I(U_i^D)$ was the node who first infected U_i^D, given all the previous infection events and the fact that U_i^D was infected at $t_{U_i^D}^D$ by one of the previously infected nodes in D. It can be obtained, according to formula 7, via:

$$p(I_i | D_{\leq i}, I_{<i}) = \frac{a_{I(U_i^D), U_i^D}^D \left(\phi_{I(U_i^D)}^D\right) / b_{I(U_i^D), U_i^D}^D \left(\phi_{I(U_i^D)}^D\right)}{\sum_{u \in \mathcal{U}, t_u^D < t_{U_i^D}^D} a_{u, U_i^D}^D \left(\phi_u^D\right) / b_{u, U_i^D}^D \left(\phi_u^D\right)} \quad (11)$$

with $I(U_i^D) \in U^D$ the infector of U_i^D stored in I.

Unfortunately the log-likelihood from formula 9 is still particularly difficult to optimize directly since it requires to consider every possible vector $I \in \mathcal{I}^D$ for each training episode D at each optimization iteration. Moreover, the probability products in formula 15 would lead to zero gradients because of decimal representation limits. Therefore, we need to define an approach where the optimization can be done via trajectory sampling. Different choices would be possible. First, Monte-Carlo approaches such as the Gibbs Sampling EM could be used, but they require to sample

from the posteriors of the full trajectories of the cascades, which is very instable and complex to perform. Another possibility is to adopt a variational approach (Kingma & Welling, 2013), where an auxiliary distribution q is learned for the inference of the latent variables. As done in (Krishnan et al., 2016) for the inference in sequences, a smoothing strategy could be developed by grounding on a bi-directional RNN that would consider past and future infections for the inference of the ancestors of nodes via $q(I_i|D, I_{<i})$ for every infected node i in an episode D. However, learning the parameters of such a distribution is particularly difficult (episodes of different lengths, cascades considered as sequences, etc.). Also, another possibility for smoothing would be to define an independent distribution q_i^D for every episode $D \in \mathcal{D}$ and every infection $i \in \{1, ..., |D|-1\}$. However, this induces a huge number of variational parameters, increasing with the size of the training set (linearly in the number of training episodes and quadratically in the size of the episodes). Thus, we propose to rather rely on the conditional distribution of ancestors given the past for sampling (i.e, $q_i^D(I_i) = p(I_i|D_{\leq i}, I_{<i})$), which corresponds to a filtering inference process.

From the Jensen inequality on concave functions, we get for a given episode D:

$$\log p(D) = \log \sum_{I \in \mathcal{I}^D} p(D, I) \geq \mathbb{E}_{I \sim q^D} \left[\log p(D, I) - \log q^D(I) \right] \triangleq \mathcal{L}(D; \Theta) \tag{12}$$

where $q^D = \prod_{i=1}^{|D|-1} p(I_i|D_{\leq i}, I_{<i})$. This leads to a lower-bound of the log-likelihood, which corresponds to an expectation from which it is easy to sample: at each new infection of a node i in a episode D, we can sample from a distribution depending on the past only which node has infected i. Maximizing this lower-bound (also called the ELBO) incitates the process to choose trajectories that explain the best the observed episode. To maximize it via stochastic optimization, we refer to the score function estimator (Ranganath et al., 2014), which leverages the the derivative of the log-function ($\nabla_\theta \log p(\mathbf{x}; \theta) = \frac{\nabla_\theta p(\mathbf{x};\theta)}{p(\mathbf{x};\theta)}$) to express the gradient as an expectation from which we can sample. Another possibility would have been to rely on the Gumbel-Softmax and the Concrete distribution with reparametrization such as done in (Maddison et al., 2016), but we observed greatly better results using the log-trick. The gradient of the ELBO function for all the episodes is given by:

$$\nabla_\Theta \mathcal{L}(\mathcal{D}; \Theta) = \sum_{D \in \mathcal{D}} \mathbb{E}_{I \sim q^D} \left[\left(\log p(D, I) - \log q^D(I) - 1 - b \right) \nabla_\Theta \log q^D(I) + \nabla_\Theta \log p(D, I) \right] \tag{13}$$

where b is a moving-average baseline of the ELBO per training episode, used to reduce the variance (taken over the 100 previous training epochs in our experiments). This stochastic gradient formulation enable to obtain unbiased steepest ascent directions despite the need to sample the ancestor vectors for the computation of the node states (with the replacement of expectations by averages over K samples for each episode). The optimization is done using the ADAM optimizer () over mini-batches of M episodes ordered by length to avoid padding ($M = 512$ and $K = 1$ in our experiments). Our full efficient algorithm is given in appendix [2].

4 EXPERIMENTS

4.1 SETUP

We perform experiments on two artificial and three real-world datasets:

- Arti1: Episodes generated on a scale-free random graph of 100 nodes. The generation process follows the CTIC model. But rather than only one transmission probability k parameter per edge, we set 5 different k_i depending on the diffusion nature. Before each simulation a number $i \in \{1, ..., 5\}$ is sampled, which determines the parameters to use. 10000 episodes for training, 5000 for validation, 5000 for testing. Mean length of the episodes=7.55;
- Digg: Data collected from the Digg stream API during a one month time window. Infections are the "diggs" posted by users on links published on the media. We kepts the 100 most active users from the collected data. 20000 episodes for training, 5000 for validation, 5000 for testing. Mean length of the episodes=4.26.

[2]The code is publicly available at XXX Anonymous.

- Weibo: Retweet cascades extracted from the Weibo microbloging website using the procedure described in (Leskovec et al., 2009). The dataset was collected by (Fu et al., 2013). 4000 nodes, 45000 episodes for training, 5000 for validation, 5000 for testing. Mean length of episodes=4.58.
- Memetracker: The memetracker dataset described in Leskovec et al. (2009) contains millions of blog posts and news articles. Each website or blog stands as a user, and we use the phrase clusters extracted by Leskovec et al. (2009) to trace the flow of information. 500 nodes, 250000 for training, 5000 for validation, 5000 for testing. Mean length of episodes=8.68. The results for this corpus are given in appendix.

We compare our model `recCTIC` to the following temporal diffusion baselines:

- CTIC: the Continuous-Time Independent Cascade model in its original version (Saito et al., 2009);
- RNN: the Recurrent Temporal Point Process model from (Du et al., 2016) where episodes are considered as sequences that can be treated with a classical RNN outputting at each step the probability distributions of the next infected node and its time-stamp;
- CYAN: Similar to `RMTPP` but with an attention mechanism to select previous states (Wang et al., 2017b);
- CYAN-cov: The same as Cyan but with a more sophisticated attention mechanism using an history of attention states, to give more weights to important nodes;
- EmbCTIC: This is a version of our model where γ enforced to equal 0, which results in ignoring recurrent states of nodes for probability computations. This corresponds to an embedded version of CTIC, similarly to the embedded version of DAIC from (Bourigault et al., 2016).

Note that to adapt baselines based on RNN for diffusion modeling and render them comparable to cascade-based ones, we add a "end node" at the end of each episode before training and testing them. In such a way, these models are able to model the end of the episodes by predicting this end node as the next event (no time-delay prediction for this node however).

We evaluate our models on two distinct tasks:

- Cascade modelling: the performances of the methods are reported in term of negative log-likelihood of the test episodes (averaged per episode). Lower values denote models that are less surprises by test episodes than others, rendering their generalization ability. The NLL measure depends on the model, but for each it renders the probability of an episode to be observed according to the model, both on which nodes are eventually infected and at what time. For our model which has to sample trajectories, the principle is the same as for learning, by considering $p(D) \approx \sum_{s=1}^{S} \frac{p(D, I^{(s)})}{q^D(I^{(s)})}$;
- Cascade generation: the models are compared on their ability to simulate realistic cascades from observed sources of diffusion. The beginning of the episode is revealed and the aim to predict the marginal probabilities of nodes to be eventually infected. The results, which correspond to averages over 1000 simulations each, are reported in term of Cross-Entropy (CE) taken over the whole set of nodes (infected or not) for each episode.

For each task we report results for different time intervals after the beginning of the episodes. 0 means nothing was observed, 1 means only the infections at the first time stamps are known, 2 means infections occurred before a delay of $maxT/10$ from the start of the episode and 3 for a delay lower than $maxT/20$.

4.2 Results

Results are given in 1 to 5. First, we observe that classical RNN based approaches have more difficulties to model test episodes than cascade based ones. This is not only true one the artificial datasets, which have been generated by the cascade model of CTIC on a graph of relationships,

NLL	0	1	2	3
rnn	36,89	36,27	23,02	16,37
cyan	37,48	36,72	22,56	15,47
cyan-cov	38,08	36,61	23,54	16,93
ctic	21,00	20,32	10,81	7,48
embCTIC	30,04	29,18	16,91	11,95
recCTIC	**19,62**	**18,90**	**9,87**	**6,87**

CE	0	1	2	3
rnn	0,63	0,60	0,41	0,30
cyan	1,27	0,81	0,47	0,32
cyan-cov	0,60	0,56	0,40	0,29
ctic	**0,44**	**0,54**	0,52	0,46
embCTIC	0,54	0,56	0,36	0,27
recCTIC	0,48	**0,54**	**0,23**	**0,16**

Table 1: Negative Log-Likelihood (NLL) and Cross Entropy of Infections (CE) on Arti.

NLL	0	1	2	3
rnn	31,03	26,18	24,15	23,02
cyan	16,82	11,56	9,32	8,03
cyan-cov	21,36	16,83	14,50	13,31
ctic	27,70	22,07	19,20	17,74
embCTIC	15,98	10,31	7,92	6,75
recCTIC	**15,67**	**10,30**	**7,86**	**6,74**

CE	0	1	2	3
rnn	0,46	0,28	0,24	0,22
cyan	0,43	**0,25**	0,21	0,19
cyan-cov	0,43	0,23	0,20	0,17
ctic	0,45	0,31	0,20	0,16
embCTIC	0,43	0,31	0,21	0,17
recCTIC	**0,43**	0,27	**0,17**	**0,14**

Table 2: Negative Log-Likelihood (NLL) and Cross Entropy of Infections (CE) on Digg.

but also on real-world datasets. This confirms the potential of such cascade-based approaches for modelling diffusion on networks. The attention mechanism of the CYAN approach allows it to get sometimes closer to the cascade based results (especially on Digg), but its performances are very variable from a dataset to another. The same observation can be made for CYAN-cov whose sophisticated attention mechanism enables to improve sometimes the results but not regularly. These methods are good for the task they were initially designed - predicting the directly next infection (this had been observed in our experimentations)-, but not for modelling purposes. This is a strong limitation since the directly next infection does not help much understanding the dynamics and predict the future of a diffusion. On the generation task, the results of the not cascade-based methods are not really better, except on the Digg and the Weibo corpora when not much of the episodes has been observed.

Our approach obtains better results than all other methods in most of settings, especially for the dynamics modelling, while prediction results are also usually good compared with its competitors. Interestingly, while embCTIC (which is also a innovation of this paper) beats usually CTIC, recCTIC often obtains even better results. This validates that the history of nodes in the diffusion has a great importance for capturing the main dynamics of the network. Our inference process looks to have well converge toward useful trajectories by leading the inference distribution to resemble to the conditionnal probability of the full ancestors vector, by our black-box inference process and the recurrent mechanism of our proposal. Except on some difficult corpora (Weibo, Memetracker) where our approach has, such as the other cascade-based models, some difficulties to predict the future from few available insights (settings 0 and 1), the approach exhibit very promising performances.

5 Conclusion

We proposed a recurrent cascade-based diffusion modeling approach, which leverages the best of both worlds between the ability of embedding the history of diffusion and the capture of the tree

NLL	0	1	2	3
rnn	27,58	28,98	18,62	17,15
cyan	29,59	30,04	30,04	18,79
cyan-cov	27,50	29,12	29,12	18,55
ctic	23,92	23,46	13,31	12,28
embCTIC	24,71	24,53	13,58	12,39
recCTIC	**21,72**	**21,36**	**11,29**	**10,34**

CE	0	1	2	3
rnn	0,59	0,37	0,30	0,28
cyan	0,59	0,37	0,31	0,29
cyan-cov	0,59	**0,36**	0,30	0,28
ctic	**0,58**	0,58	0,31	0,28
embCTIC	0,59	0,59	0,30	0,28
recCTIC	0,59	0,59	**0,21**	**0,20**

Table 3: Negative Log-Likelihood (NLL) and Cross Entropy of Infections (CE) on Weibo.

dependencies underlying the diffusion in network. Results validate the approach both for modeling and for prediction.

In this work, we rely the sampling of trajectories on a filtering approach which only rely on past observations. Outgoing works concern the development of an inductive variational distribution that rely on whole observed episodes for inference.

REFERENCES

Simon Bourigault, Sylvain Lamprier, and Patrick Gallinari. Representation learning for information diffusion through social networks: an embedded cascade model. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22-25, 2016*, pp. 573–582, 2016. doi: 10.1145/2835776.2835817. URL http://doi.acm.org/10.1145/2835776.2835817.

Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 1555–1564, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939875. URL http://doi.acm.org/10.1145/2939672.2939875.

K. Fu, Chung hong Chan, and M. Chau. Assessing censorship on microblogs in china: Discriminatory keyword analysis and the real-name registration policy. *Internet Computing, IEEE*, 17(3): 42–50, May 2013. ISSN 1089-7801. doi: 10.1109/MIC.2013.28.

Jacob Goldenberg, Barak Libai, and Eitan Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing letters*, 12(3):211–223, 2001.

Manuel Gomez-Rodriguez, David Balduzzi, and Bernhard Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pp. 561–568. ACM, 2011.

Adrien Guille, Hakim Hacid, Cecile Favre, and Djamel A. Zighed. Information diffusion in online social networks: A survey. *SIGMOD Rec.*, 42(2):17–28, July 2013. ISSN 0163-5808. doi: 10.1145/2503792.2503797. URL http://doi.acm.org/10.1145/2503792.2503797.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

R. G. Krishnan, U. Shalit, and D. Sontag. Structured Inference Networks for Nonlinear State Space Models. *ArXiv e-prints*, September 2016.

Sylvain Lamprier, Simon Bourigault, and Patrick Gallinari. Influence learning for cascade diffusion models: focus on partial orders of infections. *Social Netw. Analys. Mining*, 6(1): 93:1–93:16, 2016. doi: 10.1007/s13278-016-0406-1. URL https://doi.org/10.1007/s13278-016-0406-1.

Jure Leskovec, Lars Backstrom, and Jon Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pp. 497–506, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-495-9. doi: 10.1145/1557019.1557077.

Cheng Li, Jiaqi Ma, Xiaoxiao Guo, and Qiaozhu Mei. Deepcas: an end-to-end predictor of information cascades. *CoRR*, abs/1611.05373, 2016. URL http://arxiv.org/abs/1611.05373.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *CoRR*, abs/1611.00712, 2016. URL http://arxiv.org/abs/1611.00712.

Giang Hoang Nguyen, John Boaz Lee, Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, and Sungchul Kim. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pp. 969–976, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-5640-4. doi: 10.1145/3184558.3191526. URL https://doi.org/10.1145/3184558.3191526.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. *CoRR*, abs/1403.6652, 2014. URL http://arxiv.org/abs/1403.6652.

R. Ranganath, S. Gerrish, and D. M. Blei. Black Box Variational Inference. *ArXiv e-prints*, December 2014.

Kazumi Saito, Ryohei Nakano, and Masahiro Kimura. Prediction of information diffusion probabilities for independent cascade model. In *Proceedings of the 12th international conference on Knowledge-Based Intelligent Information and Engineering Systems, Part III*, KES '08, pp. 67–75. Springer-Verlag, 2008.

Kazumi Saito, Masahiro Kimura, Kouzou Ohara, and Hiroshi Motoda. Learning continuous-time information diffusion model for social behavioral data analysis. In *Proceedings of the 1st Asian Conference on Machine Learning: Advances in Machine Learning*, ACML '09, pp. 322–337, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-05223-1.

Yong Shi, Minglong Lei, Peng Zhang, and Lingfeng Niu. Diffusion based network embedding. *CoRR*, abs/1805.03504, 2018. URL http://arxiv.org/abs/1805.03504.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *CoRR*, abs/1503.00075, 2015. URL http://arxiv.org/abs/1503.00075.

Jia Wang, Vincent W. Zheng, Zemin Liu, and Kevin Chen-Chuan Chang. Topological recurrent neural network for diffusion prediction. *CoRR*, abs/1711.10162, 2017a. URL http://arxiv.org/abs/1711.10162.

Yongqing Wang, Huawei Shen, Shenghua Liu, Jinhua Gao, and Xueqi Cheng. Cascade dynamics modeling with attention-based recurrent neural network. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 2985–2991, 2017b. doi: 10.24963/ijcai.2017/416. URL https://doi.org/10.24963/ijcai.2017/416.

Zhitao Wang, Chengyao Chen, and Wenjie Li. Attention network for information diffusion prediction. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pp. 65–66, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-5640-4. doi: 10.1145/3184558.3186931. URL https://doi.org/10.1145/3184558.3186931.

Yuan Zhang, Tianshu Lyu, and Yan Zhang. Cosine: Community-preserving social network embedding from information diffusion cascades. In *AAAI*, 2018.

6 APPENDIX

6.1 JOINT PROBABILITY

In this section, we detail the derivation of $p(D, I)$ whose formulation is given in equation 15. For each infected node at position i, we need to compute:

- the probability for U_i^D for being infected at its time of infections given the nodes $D_{<i}$ previously infected in D and the states associated to these nodes;
- the probability of the ancestor index I_i given the i-th infection D_i, and the previous infections $D_{<i}$ associated to their states $\phi_{<i}^D$;
- the probability that not infected nodes are actually not infected by the i-th infected node given its state.

This gives:

$$
\begin{aligned}
p(D,I) = \quad & p(D_0|D_{<0},\phi^D_{<0})p(I_0|D_{\leq 0},\phi^D_{<0}) \prod_{v\notin U^D} p(v\notin U^D|D_0,\phi^D_0) \\
\times \quad & p(D_1|D_{<1},\phi^D_{<1})p(I_1|D_{\leq 1},\phi^D_{<1}) \prod_{v\notin U^D} p(v\notin U^D|D_1,\phi^D_1) \\
\times \quad & p(D_2|D_{<2},\phi^D_{<2})p(I_2|D_{\leq 2},\phi^D_{<2}) \prod_{v\notin U^D} p(v\notin U^D|D_2,\phi^D_2) \\
& \dots \\
\times \quad & p(D_{|D|-1}|D_{<|D|-1},\phi^D_{<|D|-1})p(I_{|D|-1}|D_{\leq |D|-1},\phi^D_{<|D|-1}) \\
& \prod_{v\notin U^D} p(v\notin U^D|D_{|D|-1},\phi^D_{|D|-1}) \\
= \quad & \prod_{i=1}^{|D|-1} p(D_i|D_{<i},I_{<i})\, p(I_i|D_{\leq i},I_{<i}) \prod_{v\notin U^D} p(v\notin U^D|D_{\leq |D|-1},I)
\end{aligned}
\tag{14}
$$

6.2 Generation Process

The generation process of our model is given in algorithm 1. The process iterates while there remains some nodes in a set of infectious nodes (initialized with u_0). $\oplus$ denotes the concatenation between two lists. At each iteration, the process selects the infectious node u with minimal time-stamp of infection (all time-stamps but $t^D_{u_0}$ are initialized to ∞), removes it from the infectious set and records its infector and infection time-stamp in the cascade. Then, for each node v with time-stamp greater than the one of u, u attempts to infects v according to the probability $k_{u,v}(\phi^D_u)$ (computed with eq 2). If it succeeds, v is inserted in the set of infectious nodes and a time t is sampled for v from an exponential law with parameter $r^D_{u,v}$. If the new time t for v is lower than its stored time t^D_v, this new time is stored in t^D_v, u is stored as the infector of v in the $from$ table (used to build I^D) and the new state ϕ^D_v is computed according to its new infector u. The generation process outputs a cascade structure (as described in the introduction of the previous section). From the classical `CTIC`, the only changes are at lines 12, 14 and 19, respectively for the computation of $k_{u,v}$, $r_{u,v}$ and ϕ^D_v.

6.3 Learning Process

The learning process of our model is depicted in algorithm 2. In this algorithm, the function $makeBins$ first creates minibatches by ordering $\mathcal{D}$ in decreasing length and cutting this ordered list in bins of $batchSize$ episodes each. Each bin contains 3 matrices with $batchSize$ rows (except in the last bin which contains matrices for the remaining $|\mathcal{D}|\%batchSize$ episodes):

- Inf: a matrix where the cell (i,j) contains the j-infected node in the i-th episode of the bin, or -1 if the corresponding episode contains less than j infected nodes. The width of the matrix is equal to the number of infected nodes in the longest episode in the bin (the episode in the first row of the matrix);
- $Times$: a matrix where the cell (i,j) contains the infection time-stamp of the j-infected node in the i-th episode of the bin, or -1 if the corresponding episode contains less than j infected nodes. The width of the matrix is equal to the number of infected nodes in the longest episode in the bin (the episode in the first row of the matrix);
- $NotInf$: a binary matrix with $|\mathcal{U}|$ columns where the cell (i,j) equals 0 if the node j is infected in the i-the episode of the bin, 1 otherwise;

Then, at each epoch, the algorithm iterates on every bin. For each bin, it first initializes the states of the infected nodes using a function $initStates$ which produces a tensor $\boldsymbol{\phi}$ of $nbRows(Inf)$ matrices $nbCols(Inf)\times d$ whose each row is filled by ϕ_0 (with $nbRows(X)$ and $nbCols(X)$ respectively the number of rows and columns in matrix X). For every step t of infection in the bin, the prosess first determines in $mask$ the rows of the matrices which correspond to not ended episodes ($Times[:,t]$ refers to the column t of $Times$). Then, if the step is not the initial step $t=0$, it uses functions $computeLogA$ and $computeLogB$ with nodes previously infected for each episode $Inf[mask,:t]$ associated to their corresponding states $\boldsymbol{\phi}[mask,:\,t]$. While the function $computeLogA$ returns a $nbRow(Inf[mask])\times t$ matrix where the cell (i,j) contains the log-probability for the j-th

Algorithm 1: Cascade Generation Process

Input: $\Theta, \mathcal{U}$
for $u \in \mathcal{U}$ **do**
  $t_u^D = \infty$
end
$U^D = ()$; $I^D = ()$; $t_{u_0}^D = 0$; $from_{u_0} = 0$; $Infectious = \{u_0\}$;
while $Infectious \neq \emptyset$ **do**
  $u \leftarrow \underset{x \in Infectious}{\arg\min} \; t_x^D$;
  $Infectious \leftarrow Infectious \setminus \{u\}$;
  $U^D \leftarrow U^D \oplus u$;
  $I^D \leftarrow I^D \oplus from_u$;
  for $v \in \mathcal{U} : t_v^D > t_u^D$ **do**
    $x \sim Bernouilli\,(k_{u,v}(\phi_u))$;
    if $x == 1$ **then**
      $x \sim Exp\,(r_{u,v})$;
      $t \leftarrow t_u^D + t$;
      if $t < t_v^D$ **then**
        $t_v^D \leftarrow t$;
        $from_v \leftarrow |U^D| - 1$;
        $\phi_v = \phi(\phi_u, \omega_v^{(\phi)})$;
        $Infectious \leftarrow Infectious \cup \{v\}$;
      end
    end
  end
end
Output: $C^D = (U^D, (t_u^D)_{u \in \mathcal{U}}, I^D)$;

node in the i-th episode to infect $Inf[i,t]$ at its infection time-stamp (using a matrix version of equation 4), the function $computeLogB$ returns a same shape matrix where the cell (i,j) contains the log-probability that the j-th node in the i-th episode does not infect $Inf[i,t]$ before its infection time-stamp (using a matrix version of equation 5). Then, it computes the log-probability for each infected at step t to be actually infected at their time-stamp of infection (line 15, which corresponds to a matrix computation of eq 7 for the infected nodes at step t). $logsumexp(X,1)$ is a function which returns the logarithm of the sum of the exponential values from each row of a matrix X and $sum(X,1)$ is a function which returns the vector of the sums of each row from X .

Then, there are some choices to be made about ancestors of infected nodes. $logPi$ is matrix whose each cell (i,j) contains the log-probability that the infected node at step t in the episode of row i was infected by the j-th infected node in this episode. Ancestors at step t are sampled from categorial distributions parameterized by the $logPi$ matrix, and the correspond sample log-probabilities are added to $logq$.

Now that the ancestors are sampled for the infected nodes at step t, we can use them to compute their states. The function $computeStates$ is a matrix version of equation 1 which computes new states for the nodes infected at step t according to the states of the sampled ancestors in $\boldsymbol{u}$.

At the end of each iteration t, the log-likelihood that not infected nodes in $NotInf[mask]$ are actually not infected by infected nodes at step t is computed via $computeLogG$, which is a matrix version of equation 8. This quantity is added to the accumulator ll.

At the end of the bin (when $t == nbCols$), a control variate baseline is computed by maintaining a list bh of the quantity vectors considered in $\nabla_\Theta \mathcal{L}(\mathcal{D})$. The baseline b considered in the stochastic gradient for any episode D is thus equal to the average of $(logp(D) - 1)$ for this specific episode taken over the b_length previous epochs.

Finally, the gradients are computed and the optimizer ADAM is used to update the parameters of the model.

Algorithm 2: Learning Process

Input: $\mathcal{D}, \mathcal{U}$, $batchSize$, $nbEpochs$, Θ, b_length
$bins \leftarrow makeBins(\mathcal{D}, \mathcal{U}, batchSize)$;
for $epoch \in \{1, ..., nbEpochs\}$ **do**
  $ibin \leftarrow 0$;
  for *(Inf,Times,NotInf) in bins* **do**
    $ll \leftarrow (0)_{nbRows(Inf)}$; $logq \leftarrow (0)_{nbRows(Inf)}$;
    $\boldsymbol{\phi} \leftarrow initStates(\phi_0, nbRows(Inf), nbCols(Inf))$;
    for $t \in \{0, ..., nbCols(Inf)\}$ **do**
      $mask \leftarrow (Times[:, t] >= 0)$;
      if $t > 0$ **then**
        $A \leftarrow computeLogA(\boldsymbol{\phi}[mask, : t], Inf[mask, : t], Inf[mask, t])$;
        $B \leftarrow computeLogB(\boldsymbol{\phi}[mask, : t], Inf[mask, : t], Inf[mask, t])$;

        # Compute $P(D_i, I_i | D_{<i}, I_{<i})$
        $H \leftarrow logsumexp(A - B, 1) + sum(B, 1)$;
        $ll[mask] \leftarrow ll[mask] + H$;

        # Sample from $P(I_i | D_{\geq i}, I_{<i})$
        $logPi \leftarrow (A - B) - logsumexp(A - B)$;
        $\boldsymbol{u} \sim Categorical(exp(logPi))$;
        $logq[mask] \leftarrow logq[mask] + logPi[mask, \boldsymbol{u}]$

        $\boldsymbol{\phi}[mask, t] \leftarrow computeStates(\boldsymbol{u}, \boldsymbol{\phi}[mask, : t], Inf[mask, : t], Inf[mask, t])$;
      end
      $ll[mask] \leftarrow ll[mask] + computeLogG(\boldsymbol{\phi}[mask, t], Inf[mask, t], NotInf[mask])$;
    end
    $bh[ibin] \leftarrow h[ibin] \oplus (ll - logq - 1)$;
    if $epoch \geq b_length$ **then**
      $bh[ibin].pop(0)$;
    end
    $b \leftarrow sum(bh[ibin], 1)/min(epoch + 1, b_length)$;
    $\nabla_\Theta \mathcal{L}(\mathcal{D}; \Theta) \leftarrow \frac{1}{nbRows(Inf)} [sum((ll - logqz - 1 - b) \nabla_\Theta logq) + \nabla_\Theta \log ll]$;

    $(\Theta) \leftarrow ADAM(\nabla_\Theta \mathcal{L}(\mathcal{D}; \Theta))$;

    $ibin \leftarrow ibin + 1$;
  end
end

6.4 Results on the MemeTracker Corpus

NLL	0	1	2	3
rnn	112,35	122,12	110,48	103,61
cyan	115,23	115,23	109,20	102,10
cyan-cov	95,58	95,58	95,58	90,20
ctic	52,70	52,61	48,48	44,33
embCTIC	54,18	53,37	49,68	45,15
recCTIC	**50,11**	**50,38**	**48,35**	**42,20**

CE	0	1	2	3
rnn	1,68	1,68	1,59	1,51
cyan	1,66	1,66	1,59	1,49
cyan-cov	1,61	**1,61**	1,52	1,39
ctic	**1,33**	1,70	1,60	1,46
embCTIC	1,59	1,69	1,57	1,39
recCTIC	1,51	1,62	**1,49**	**1,36**

Table 4: Negative Log-Likelihood (NLL) and Cross Entropy of Infections (CE) on Memetracker.