Predicting Temporal Activation Patterns via Recurrent Neural Networks

Giuseppe Manco, Giuseppe Pirrò and Ettore Ritacco

ICAR - CNR, via Pietro Bucci 8/9C, 87036 Arcavacata di Rende (CS), Italy,
{name.surname@icar.cnr.it}

Abstract. We tackle the problem of predict whether a target user (or group of users) will be active within an event stream before a time horizon. Our solution, called PATH, leverages recurrent neural networks to learn an embedding of the past events. The embedding allows to capture influence and susceptibility between users and places closer (the representation of) users that frequently get active in different event streams within a small time interval. We conduct an experimental evaluation on real world data and compare our approach with related work.

1 Introduction

There is an increasing amount of streaming data in the form of sequences of events characterized by the time in which they occur and their mark. This general model has instantiations in many contexts, from sequences of tweets characterized by a (re)tweet time and identity of the (re)tweeter and/or the topic of the tweet, to sequences of locations characterized by the time and location of each check-in. We focus on influence-based activation networks, that is, event sequences where the occurrence of an event can boost or prevent the occurrence of another event. Understanding the structural properties of these networks can provide insights on the complex patterns that govern the underlying evolution process and help to forecast future events.

The problem of inferring the topical, temporal and network properties characterizing an observed set of events is complicated by the fact that, typically, the factors governing the influence of activations and their dependency from times are hidden. Indeed, we only observe activation times (e.g. retweet time) and related marks, while, activations can depend on several factors including the stimulus provided by the ego-network of a user or his attention/propensity towards specific themes.

The goal of this paper is to introduce PATH (Predict User Activation from a Horizon), which focuses on scenarios where there is the need to *predict whether a target user (or group of users) will be active before a time horizon* T_h. PATH can be used, for instance, in market campaigns where target users are the potential influencers that if active, before T_h, can contribute to further spread an advertisement and trigger the activation of influencees that can be made aware of a certain product/service. PATH learns an embedding of the past event history via Recurrent Neural Networks that also cater for the diffusion memory.

The embedding allows to capture influence and susceptibility between users and places closer (the representation of) users that frequently get active in different streams within a small time interval.

1.1 Related Work

A number of proposals have addressed the problem of modeling streams of events via neural networks: (i) approaches like DeepCas [7] and DeepHawkes [2] tackle the problem of predicting the length that a cascade will reach within a timeframe or its incremental popularity; (ii) approaches like Du et al. [3] and Neural Hawkes Process (NHP) [8] model and predict time event markers and time; (iii) Survival Factorization (SF) [1] leverages influence and susceptibility for time and event predictions. `PATH` adopts a different departure point from these approaches: it focuses on predicting the activation of (groups of) users before a time horizon instead of the exact activation time.

Differently from (i) `PATH` considers time and uses an embedding to capture both influence and susceptibility between users and predict future activations. Moreover, (i) focuses on the prediction of cumulative values only (e.g., cascade size). Differently from (ii), we do not assume that time and event are independent and capture their interdependencies via the embedding and cascade history. Besides, (ii) focuses on predicting event types only (e.g., popular users), which is not enough in the scenarios targeted by `PATH` (e.g., targeted market campaigns) where one is interested in predicting the behavior of specific users instead of their types. A for (iii), it fails in capturing the cumulative effect of history while `PATH` captures by using an embedding.

The contributions of the paper are as follows: **(i)** `PATH`, a classification-based approach based on recurrent neural networks allowing to model the likelihood of observing an event as a combined result of the influence of other events; **(ii)** an experimental evaluation and a comparison with related work.

The remainder of the paper is organized as follows. We introduce the problem in Section 2. We present `PATH` in Section 3. We compare our approach with related research in Section 4. We conclude and sketch future work in Section 5.

2 Problem Definition

We focus on network of individuals who react to solicitations along a timeline. An activation network can be viewed as an instance of a marked point process (e.g., information cascades) on the timeline, defined as a set $\mathcal{H} = \{\mathcal{H}^c\}_{1 \leq c \leq m}$. Here, $\mathcal{H}^c = \{(t_i, u_i)\}_{1 \leq i \leq m_c}$ represents a cascade where $u_i \in \mathcal{U}^c$ is a user in a set of all users that get active in the cascade $\mathcal{H}^c$ at time $t_i \in \mathbf{t}^c$ where $\mathbf{t}^c$ is the projection over the timestamps in $\mathcal{H}^c$. We denote by $\mathcal{U}$ the set of users in all cascades.

Given a cascade $\mathcal{H}^c$, we denote by $\mathcal{H}^c_{<t}$ (resp. $\mathcal{H}^c_{\leq t}$) the set of events $e_i \in \mathcal{H}^c$ such that $t_i < t$ (resp., $t_i \leq t$). The terms $t^c_{<t}$and $\mathcal{U}^c_{<t}$ can be defined accordingly.

2.1 Modeling diffusion

We start from the observation that what is likely to happen in the future (viz. which user will be active and when) depends on what happened in the past (viz. the chain of previously active users). One important point to take into account is the susceptibility of users, that is, the extent to which they are influenced by specific previously activated users. Our model should be flexible enough to reflect both *exciting* and *inhibitory* effects. While the former boosts the likelihood of observing u active in c, the latter actually could prevent it to do so.

Given a cascade $\mathcal{H}^c$, a timestamp $t \geq 0$ and a user $u \notin \mathcal{U}^c_{<t}$, the goal is to obtain an estimate of the density function $f(t, u|\mathcal{H}^c_{<t})$, which can be used to model the following evolution scenario: *given a cascade c and time horizon T^c_h; how likely is it that u will become active in c within T^c_h?*
The challenge, at this point, is how to concretely formulate the density f. We can decouple its specification as follows:

$$f(t, u|\mathcal{H}^c_{<t}) = g(t|u, \mathcal{H}^c_{<t}) \cdot h(u|\mathcal{H}^c_{<t}), \tag{2.1}$$

where the first component represents the likelihood that u becomes active within t, given $\mathcal{H}^c_{<t}$, and the second component represents the likelihood that u activates (independent of the time) as a reaction to the current history.

The modeling of $\mathcal{H}^c_{<t}$ can include different pieces of information, among which, the sequence of user activations, their activation times, the relative activation speed, and possibly the topic of the cascade. Nevertheless, our assumption is that $\mathcal{H}^c_{<t}$ can also encode latent information including susceptibility and influence between users that can be derived, for instance, from neighborhood information in a networks (e.g., follower/followee relations in Twitter) or user behaviors (e.g., users that retweet after a certain set of other influential users (re)tweet). This is exactly what we want to unveil in our modeling.

2.2 Embedding history

We want to learn and embedding of users in a latent K-dimensional space such that users in the same cascade are closer in the embedding, and users within different cascades are distant.

The embedding is based on the idea of a virtual graph $G = (\mathcal{V}, \mathcal{E})$ where $(u, v) \in \mathcal{E}$ if and only if there is a cascade c such that $v \prec_c u$ (viz. u precedes v in the cascade c). Thus, the influence exerted by v on u represents the probability that, any random walk on G starting from v, eventually touches u. By assuming that $\alpha_{u,v}$ represents such a probability, we factorize it as $\alpha_{u,v} = \mathbf{s}_u \cdot \mathbf{a}_v$, where $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_N], \mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_N] \in \mathbb{R}^{N \times K}$ are the susceptibility and influence matrices, respectively. Matrices are computed by relying on the standard network architecture borrowed from the `word2vec` paradigm [9]:

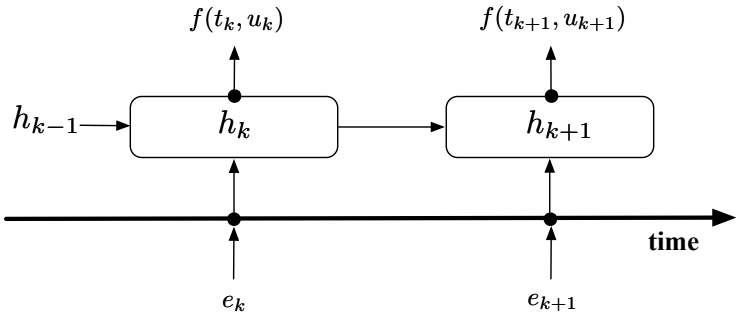
$$\mathbf{a}_k = \mathbf{W}_e \mathbf{u}_k \qquad \mathbf{s}_k = \mathbf{V}_e \mathbf{u}_k$$

Here, $\mathbf{u}, \mathbf{v}$ represents the one-hot encodings of u and v and $k \in [1, ..., K]$. The matrices $\mathbf{W}_e, \mathbf{V}_e$ represent the embeddings, obtained by minimizing an adapted

form of contrastive loss [4] that penalizes the distance of users within the same cascades and the closeness of users in different cascades.

2.3 Capturing the diffusion memory

To encode temporal relationship within $\mathcal{H}^c_{<t}$ we use recurrent neural networks (RNNs). An RNN is a recursive structure that, at the current step, gets as input the previous network state (the outputs form the hidden units) along with the current input to compute a new state. The following picture provides an overview of a simple RNN cast to our context.



At each step k, we feed into the network an event $(t_k, u_k) \in \mathcal{H}^c$ that encodes the current user (u_k) and its activation time (t_k). The learned hidden state (h_k) represents the non-linear dependency between these components and past events, which can be used to model $f(t_k, u_k|\mathcal{H}^c_{<t_k})$. In the following, we adopt the LSTM instantiation of the RNN framework [5]. The idea of an LSTM unit is to reliably transmitting important information many time steps into the future. At every time step, the unit modifies the internal status by deciding which part to keep or replace with new information coming from the current input. In the following we shall use the shortcut $\mathbf{h}_k = \text{LSTM}(\mathbf{z}_k, \mathbf{h}_{k-1})$ to denote a basic functional architecture that elaborates an input $\mathbf{z}_k$ and outputs the updated state.

3 PATH: Predicting User Activation from a Horizon

We now introduce `PATH` (Predicting User Activation from a Horizon), which focuses on simplifying $f(t, u|\mathcal{H}^c_{<t})$ as the binary response function $\mathbb{I}(t \leq T_h|u, \mathcal{H}^c_{<t})$ that denotes whether u becomes active in c within T_h. We focus on events $(t_k, u_k) \in \mathcal{H}^c$ and consider as an additional feature the time delay $\delta_k = t_k - t_{k-1}$ relative to the previous activation within the cascade. This allows us to capture the property that cascades may have intrinsically different diffusion speeds causing some of them to concentrate users' activations in a short timeframe while

others in a more extended interval. In what follows we consider a cascade $\mathcal{H}^c$ enhanced with information about time delays, that is, $\mathcal{H}^c = \{(t_k, u_k, \delta_k)_{1 \leq k \leq m_c}$. Given a partially observed cascade $\mathcal{H}^c_{<t_l}$ (with $t_l < T^c_h$ representing the timespan of the observation window), our objective is to predict, for a given entity $u \notin \mathcal{U}^c_{t_l}$, whether $u \in \mathcal{U}^c_{T^c_h}$. In order to uncover all the characteristics of the activations within cascades, we consider a model built on all possible prefixes of the available cascades.

In other words, for a given cascade c of length m_c, we feed into the network cascade prefixes starting from 2 elements until reaching $m_c - 1$ elements. Notice that, we do not consider the 1 element prefix, which we assume becomes "spontaneously" active. Moreover, we also add negative examples as follows: for each $u \notin \mathcal{U}^c$, we associate the cascades $\mathcal{H}^c_{\leq t_j} \cup \{(t_j, u, \delta_j)\}$ (with $1 \leq j \leq m_c - 1$) and $\mathcal{H}^c \cup \{(T^c_h, u, (T^c_h - t_{m_c}))\}$ with negative labels. Again, the intuition is that, since u is not active no partial cascade provides the sufficient intensity to activate u within the given time horizon.

Adding negative examples in the data preparation represent an effective data augmentation process, which enlarges the training data by inferring new inputs in the training set. This is crucial to let the approach better fine tune separation between active and inactive users, as well as better characterize the true activation time of active users. Let $\mathcal{T}_\mathcal{C}$ denote the set of all pairs $\langle \mathcal{H}^c_{\leq t_i}, y_i \rangle$ that can be built as described above. Our idea is to exploit the embedding and LSTM tools described in the previous section to solve the supervised problem at hand.

Figure 1 illustrates the basic architecture of the model where arrows represent inputs and boxes the elements of the architecture. The input layer (bottom part) takes as input the user for which we want to predict the activation (u_n) along with information about already active users represented by triples of the form of (u_k, t_k, δ_k) and the network history (h_k) (recurrent layer in the figure).

Given a pair $\langle \mathcal{H}^c_{\leq t_i}, y_i \rangle \in \mathcal{T}_\mathcal{C}$ with $|\mathcal{H}^c_{\leq t_i}| = n$ and by considering $(t_k, u_k, \delta_k) \in \mathcal{H}^c_{<t_i}$ (with $1 \leq k \leq n$), the architecture of the network can be captured by the following equations:

$$\mathbf{a}_k = \mathbf{W}_e \mathbf{u}_k \tag{3.1}$$

$$\mathbf{h}_k = \text{LSTM}\left([\mathbf{a}_k, t_k, \delta_k], \mathbf{h}_{k-1}\right) \tag{3.2}$$

$$\hat{y}_i = \sigma\left(\mathbf{W}_o \mathbf{h}_n\right) \tag{3.3}$$

$$\tilde{y}_i = \exp\left\{ - \left\| \mathbf{a}_n - \sum_{k=1}^{n-1} \mathbf{a}_k \right\|^2 \right\} \tag{3.4}$$

In the output layer, $\hat{y}_i$ represents the probability that y_i is positive, as provided by the network: that is, it encodes the probability that u_n becomes active within t_n; $\tilde{y}_i$ encodes the affinity between u_n and all users preceding it within $\mathcal{H}^c_{\leq t_i}$. The distance $\|\mathbf{a}_n - \sum_{k=1}^{n-1} \mathbf{a}_k\|$ plays a crucial role here: since the target user is on the tail of the cascade, the embedding should emphasize the similarities with the predecessors that trigger an activation, and by the converse minimize the similarities with those ones which do not trigger it.

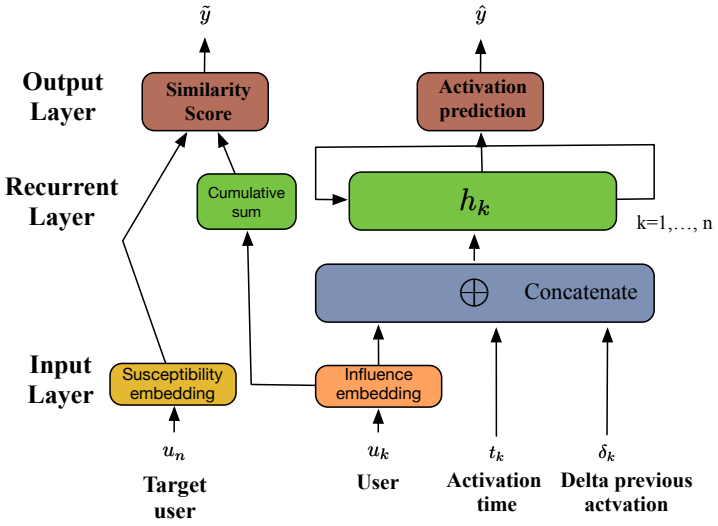


Fig. 1: Overview of the architecture of PATH.

The loss is a combination of cross-entropy and the embedding loss previously described:

$$\mathcal{L} = \sum_{\substack{\langle \mathcal{H}, y \rangle \in \mathcal{T}_C \\ |\mathcal{H}|=n}} \{ y \left(\gamma \log(\hat{y}) + \beta \log \tilde{y} \right) + (1-y) \left(\gamma \log(1-\hat{y}) + \beta \log(1-\tilde{y}) \right) \} \tag{3.5}$$

where γ and β are weights balancing cross-entropy and embedding.

4 Experiments

We validate our approach by analysing the algorithm on real-life datasets. In particular, we analyse the capability of the algorithm at predicting the activation time of users within an information cascade. The implementation we use in the experiments can be found at `https://github.com/gmanco/PATH`.

4.1 Datasets

We evaluated PATH by exploiting two real-world datasets containing propagation cascades crawled from the timelines of `Twitter`[1] and `Flixster`.[2] In particular, `Twitter` includes ∼32K nodes with ∼9K cascades while `Flixster` includes

[1] `http://www.twitter.com/`
[2] `http://www.flixster.com/`

∼2K nodes with ∼5K cascades. The propagation mechanism on `Twitter` is expressed by retweeting, in other words a chain of repetitions and transmissions of a tweet from a set of users to their neighbors in a recursive process. Each activation corresponds to a retweet. An activation in `Flixster` happens when a user rates a movie, while a cascade is composed by all the activations related to the same movie.

The two datasets differ essentially for the following characteristics: `Twitter` includes a larger number of users and shorter delays than `Flixster`. In addition, retweets intuitively highlight two relevant aspects, namely the importance of the topic and the single influence of the individual from which the retweet is performed. By contrast, movie ratings are more likely to exhibit a cumulative effect: popular movies are more likely to be considered than unpopular ones.

4.2 Evaluation Methodology.

We evaluate `PATH` against two baseline models, both relying on Survival Analysis [6]. The first instantiation implements a Cox proportional hazard model (*CoxPh* in the following). We implement the model using the `lifelines`[3] package and extract, for each event $(t_k, u_k, \delta_k) \in \mathcal{H}^c$, the following features: (1) size of the prefix; (2) last activation time; (3) average delay for each active user so far; (4) number of neighbors in the history, and (5) coverage percentage of them within the history; (6) the activation time of the most recent neighbor, if any; (7) correlation between the activation of the current user an its neighbors within the history, computed in previous cascades. This model represents an intuitive baseline where features are manually engineered and include a mix of external information (coming from the underlying network neighborhood) and information derived from the cascade itself.

The second instantiation is given by the *Survival Factorization* (*SF* in the following) framework described in [1]. The comparison is important since *SF* relies on the same guiding ideas of `PATH` (influence and susceptibility) with the substantial difference that there is no cumulative effect of $\mathcal{H}^c_{<t_k}$, but instead an influential user has to be detected for each activation.

To evaluate the approaches, we proceed as follows: given training and test sets $\mathcal{C}_{train}$ and $\mathcal{C}_{test}$, we train the model on $\mathcal{C}_{train}$ and measure the accuracy of the predictions on $\mathcal{C}_{test}$. The two sets are obtained by randomly splitting the original dataset by ensuring that there is no overlap among the cascades of the two sets, but there is no entity in the test that has not been observed in the training. We used 70% of data for training and 30% for testing.

For the evaluation, we chronologically split each cascade $c \in \mathcal{C}_{test}$ into c_1 and c_2 such that, for each $u \in c_1$ and $v \in c_2$, we have that $u \prec_c v$. Next, we pick a random subsample $c_3 \subseteq \mathcal{U} - \mathcal{U}^c$. Then, given a target horizon T^c_h, we measure *TP*, *FP*, *TN* and *FN* by feeding the models on c_1 and then predicting the activation within T^c_h for each element in $c_2 \cup c_3$.

[3] see `http://lifelines.readthedocs.io` for details.

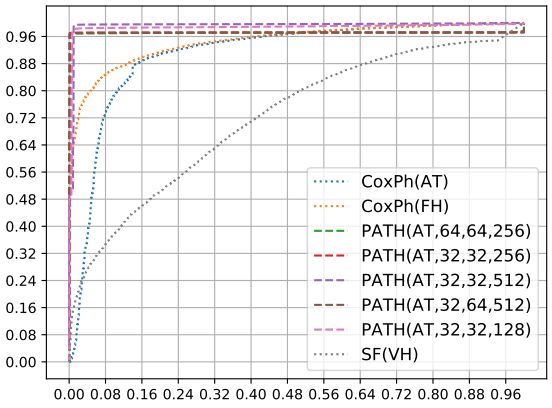


Fig. 2: ROC Curves for `PATH`,*CoxPh* and *SF* on `Flixster`.

The choice of T_h^c can follow different strategies; Fixed horizon (Fixed horizon (FH): setting T_c^h as the maximum observed activation time $T_h^{test} = \max\{t | t \in \mathbf{t}^c, c \in \mathcal{C}_{test}\}$; Variable horizon (**VH**): varying T_h^c from the smallest to the largest activation time and computing the activation probabilities associated to each possible value; Actual Time (**AT**): a particular case of the VH strategy, where $T_h^c \triangleq T_h^{u,c}$ is relative to the true activation time in c of each user $u \in c_2 \cup c_3$.

In the experiments, we plot the ROC and the F-Measure curves relative to the above alternatives and report the AUC and F values. Notice that, for *PATH*, the encoding of sequences as described in section 3 already presumes that users are evaluated on intermediate timestamps prior to their actual activation. Thus, VH and AT roughly coincide in this case. Since both *CoxPh* and *SF* are capable of inferring, for each $(u, \mathcal{H})$ pair, the probability $S_u(t|\mathcal{H}^c)$, the comparison with `PATH` is accomplished by computing $1 - S_u(\tilde{t}|\mathcal{H}^c)$ where $\tilde{t}$ is the above described horizon timestamp.

The parameter space for `PATH` was explored by grid-search, measuring the loss on a on a separate portion of the training set by 5-fold cross-validation. In the following we report 5 different instantiations, which differ from the number of cells in the LSTM (32/64), the dimensionality of the embedding (32/64) and the batch size in the training (128/256/512). Concerning *SF*, the number of factors was set to 16 for both datasets.

4.3 Evaluation Results

Figure 2 and Figure 3 report the ROC curves for the experiments. We can observe that `PATH` consistently outperforms the baselines and in particular exhibits a very good accuracy on all configurations. This is especially true on `Flixster`, where by the converse *SF* does not seem capable of correctly correlating previous

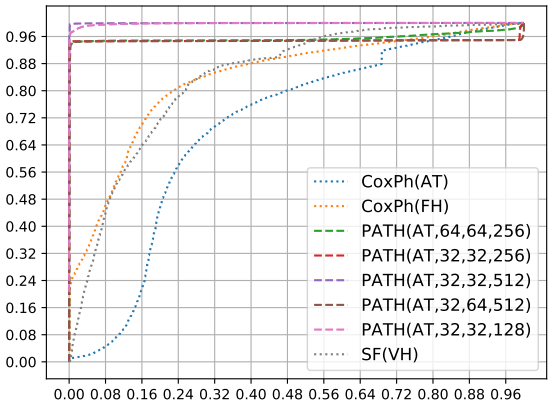


Fig. 3: ROC Curves for `PATH`,*CoxPh* and *SF* on `Twitter`.

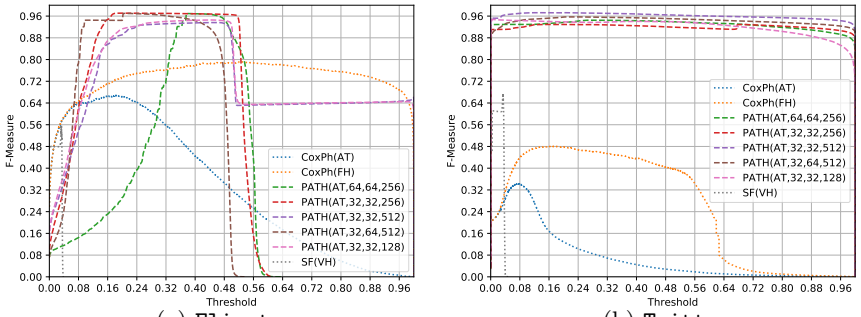
activations times. The cumulative influence effect is evident here, as a natural consequence of the underlying domain where cascading effects are more likely as a consequence of a "word of mouth" process. On `Twitter`, where the activation is more likely due to the influence of a single user (as testified by the good performance of *SF*), `PATH` still achieves the best scores, thus proving the capability of the recurrent layer to adapt the influence to a single user.

Figure 4 (a) and (b) display the F-measure curves for varying values of the threshold on the probabilities. Here, we can observe that, contrary to the baselines, higher thresholds do not cause a significant drop of the recall. The only exception is *CoxPh* (FH), which seems more stable on `Flixster`. This is a clear sign that the probabilities associated with active and inactive users in `PATH` differ substantially, and in particular active events are associated with significantly higher probabilities than inactive events.

5 Concluding Remarks and Future Work

We focused on the problem of predicting user activations in a given time horizon and show that the embedding of the user activation history, where users that become active on the same cascades are placed close, can be effectively learned via recurrent neural networks.

Experiments performed on real datasets show the effectiveness of the approach in accurately predicting next activations. In particular, it emerged that that focusing on a time horizon is more effective than predicting actual activation times, and that the proposed network architecture successfully captures the resulting classification capabilities. The network is capable of summarizing both cumulative and separate influential effects, and moreover can predict based on the properties based on the history.



(a) `Flixster` (b) `Twitter`

Fig. 4: F-Measure curves for `PATH`,*CoxPh* and *SF* on both datasets.

It is natural to wonder whether it is possible to cast the intuitions behind our approach in a generative setting, to predict both which user is likely to become active, and the time segment upon which s/he will become active. It is also natural to wonder whether the basic ideas of survival analysis can be cast within our framework, and whether it is possible to recover the distinction between influence and susceptibility in the embedding proposed in the model.

References

1. N. Barbieri, G. Manco, and E. Ritacco. Survival factorization on diffusion networks. In *ECML PKDD*, pages 684–700, 2017.
2. Q. Cao et al. Deephawkes: Bridging the gap between prediction and understanding of information cascades. In *ACM CIKM*, pages 1149–1158, 2017.
3. N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song. Recurrent marked temporal point processes: Embedding event history to vector. In *ACM SIGKDD*, pages 1555–1564, 2016.
4. R. Hadsell, S. Chopra., and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, pages 1735–1742, 2006.
5. S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
6. J.D. Kalbfleisch and L.P. Ross. *The Statistical Analysis of Failure Time Data*. Wiley Series in Probability and Statistics, 2002.
7. C. Li, J. Ma, X. Guo, and Q. Mei. Deepcas: An end-to-end predictor of information cascades. In *WWW*, pages 577–586, 2017.
8. H Mei and J.M. Eisner. The neural hawkes process: A neurally self-modulating multivariate point process. In *NIPS*, pages 6757–6767, 2017.
9. T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.