Representation Learning for Information Diffusion through Social Networks: an Embedded Cascade Model

Conference Paper · February 2016

DOI: 10.1145/2835776.2835817

Sylvain Lamprier
Sorbonne Université
62 PUBLICATIONS **360** CITATIONS

Representation Learning for Information Diffusion through Social Networks: an Embedded Cascade Model

Simon Bourigault, Sylvain Lamprier, Patrick Gallinari

Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France
CNRS, UMR 7606, LIP6, F-75005, Paris, France

firstname.lastname@lip6.fr

ABSTRACT

In this paper, we focus on information diffusion through social networks. Based on the well-known Independent Cascade model, we embed users of the social network in a latent space to extract more robust diffusion probabilities than those defined by classical graphical learning approaches. Better generalization abilities provided by the use of such a projection space allows our approach to present good performances on various real-world datasets, for both diffusion prediction and influence relationships inference tasks. Additionally, the use of a projection space enables our model to deal with larger social networks.

Keywords

Machine learning; Information diffusion; Representation Learning

1. INTRODUCTION

Over the last 10 years, the sharing of content through Online Social Networks has become a major source of information for many users. Most websites now use social networks APIs that enable users to easily retweet, share, rate, comment, tag or like any piece of information. User actions are a rich source of information for companies and for marketing. They can for example be used as indicators of *popularity* for items or persons and predicting the popularity of information items has been widely studied [20] by analyzing whole *population* statistics. Many problems in domains like information propagation, social network analysis or marketing require a finer grained modeling of diffusion and propagation phenomena, at the scale of *users*. With the development of social resources, an increasing attention has been devoted to this modeling problem. For web and social data, analyzing user interactions for modeling diffusion phenomena presents a series of difficulties:

- Users are often heterogeneous with different behaviors or interests and they interact with each other through various channels at the same time.
- Relationships between users are hard to detect and to characterize. For example, it has been shown that "weak ties" between users who don't often interact with each other are important in the information diffusion process [2], whereas they are difficult to capture.
- Diffusion cascades vary in length considerably in different applications, which makes their dynamics difficult to learn and to predict [10].

The observation of a diffusion process often comes down to collecting interaction traces indicating when people do post or send a piece of information or buy a product. Most often, the process underlying the observations is hidden: one just observes when and where the actions do occur but one does not know why. Modeling the diffusion process then amounts to approximate this mechanism based on the observations. The model can then be used in different tasks relevant for the specific application like predicting who will get infected, inferring the diffusion structure, detecting influential users, etc. The purpose of this paper is to develop a diffusion model by learning from observations. For this, we will focus on the well-known Independent Cascade model (IC) which defines an iterative process of diffusion in a network. In this context, the problem of modeling diffusion comes down to learning probability distributions characterizing the hidden influence relationships between users, in order to discover the main communication channels of the network. Learning diffusion distributions in IC from observation data has been initiated in the mid 2000 [18, 34] and since that several variants have been proposed as described in section 2. The models that we consider here perform the following hypothesis: infection[1] is binary, the diffusion network is unknown, influence relationships do not depend on the propagated content (while this could be easily incorporated in our model), infection probabilities between users do not vary in time.

In existing cascade models, diffusion probabilities are learned using non parametric or parametric formulations of the distributions defined on the edges of a diffusion graph. We develop here an alternative approach based on representation learning techniques. Instead of learning probabilities on a discrete structure corresponding to a diffusion

WSDM'16, February 22–25, 2016, San Francisco, CA, USA.
© 2015 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-3716-8/16/02…$15.00
DOI: http://dx.doi.org/10.1145/2835776.2835817

[1]Throughout this paper, we indifferently talk about infection or contamination to denote the fact that the propagated content has reached a given user of the network.

graph, we model the diffusion space as a continuous latent space where relative positions of users are used to define their content transmission likelihood. We follow the general scheme defined in [34] for learning IC distributions and propose a diffusion model based on users embeddings in a latent projection space. This model presents better generalization abilities with a lower memory cost. Our contributions are summarized below.

- We propose a new model for learning diffusion probabilities from diffusion episode observations. Its novelty consists of mapping the observation episodes onto a continuous latent space. The probabilities will be inferred from the relative positions of the users in this space. This model presents several advantages: 1. it allows us to learn directly from the observed data without making hypothesis about an underlying network which is very often only partially known. 2. this non parametric formulation allows to adjust the model complexity to the dynamics of the observed data so that the model offers a better control over generalization ability. 3. the model allows us capturing dependencies between probabilities which are not considered with other IC models. 4. the computational memory requirements are greatly reduced w.r.t. discrete based diffusion models, which allows us to consider more complex interaction schemes.

- We develop an algorithm based on a stochastic gradient approach which is well suited to handle large observation sets.

- We evaluate the algorithm on two tasks: contamination prediction in order to retrieve the eventually infected users and influence relationships inference, and compare it with the baseline IC model of [34] and with two state of the art models.

The paper is organized as follows. Section 2 presents background and motivations for the proposed approach. Section 3 defines the notations used throughout the paper and describes the diffusion model. Section 4 presents evaluation of the model on real datasets for the two tasks of infection prediction and link prediction.Finally, Section 5 concludes our works and gives some insights into possible future works.

2. MOTIVATIONS AND RELATED WORKS

While the study of diffusion phenomena firstly emerged in epidemiology and social science contexts, the development of social networks opened an unprecedented era for new related research directions. Most of the proposed iterative models of diffusion are based on two fundamental models (that can be unified on the same framework [21]) : the Independent Cascade (IC) [13] model and the Linear Threshold (LT) [16] model. Both model a user-to-user contamination process : while IC models the spread of diffusion as cascades of infections over the network, LT determines infections of users according to thresholds of the influence pressure incoming from the neighborhood of each user (assuming influence additivity). We focus in this paper on IC-like formalisms, or more generaly on cascade-based approaches which appear well fitted to reproduce realistic diffusion dynamics [20]. While the parameters of these models (transmission probabilities) initially needed to be set manually, Gruhl et al. [18] developed a first attempt to automatically learn diffusion probabilities from observations. A few years later, in a seminal paper, [34] proposed the learning methodology we extend in this article. It corresponds to a clear improvement w.r.t. [18], since it replaced an "exactly one influencer" assumption by a more realistic "at least one influencer" one. From there, IC has served as a basis for many variants and extensions of cascades-based approaches. Several of these variants consider different ways for dealing with users' infection timestamps [32, 33, 14, 12], others focus on the inclusion of context information in the original IC model [35, 38, 19, 22, 36].

Cascade based models classically make the assumption that the diffused content transits from users to users in the social network following explicit links in the graph[34, 39, 14, 37]. This is inherited from historical models of diffusion and is not adapted to many internet situations. For example, it is often the case that a user does not cite his source (blogs, tweet) and in this case there is no indication of who infected him. The social network is generally not a closed world and users may get their information from many different channels - again there is no way to go back to the source or to know how long it took for propagating the information. In other domains like in marketing, one can track the actions of a customer, but one does not know who influenced him. To summarize, even though the links of a network may sometimes correspond to useful information, they are often not representative of the real diffusion channels of a social network [39, 28]. In addition to the above examples, note that the social graph captured on an online social network (friends, followers, subscriptions...) is often incomplete, irrelevant [37] or sometimes unknown (for privacy reasons for instance). Cascade models can be used when there is no a priori graph structure, simply by considering a complete graph linking or by considering an interaction graph including all possible transmissions observed in a training set [2]. Either one of the two solutions has been used in several works like [28, 15, 14]. Note that [15] and [29] also proposed methods for efficiently selecting relevant edges from the complete graph of users.

However, considering complete graph structures (or rather dense interaction graphs) appears intractable for many datasets: the number of parameters of IC is $\mathcal{O}(d \times |\mathcal{U}|)$, with $|\mathcal{U}|$ the number of users in the network and d its average degree. Its memory complexity very quickly grows with the size of the network. By transposing the problem in a continuous space, our model skips this limitation of discrete formulations. Compression abilities offered by the representation learning field help to cope with such complexity.

Besides, the diffusion dynamics are usually influenced by many factors that are difficult to model a priori. Classical cascade models have difficulties to reproduce them. Social graphs are indeed known to possess lots of properties that distinguish them from random graphs (power-law node degrees, low diameter, etc...) [27]. One important and well-known property of social networks is that they contain *communities*, which can roughly be defined as groups of "similar" users. Recently, [3] showed that when dealing with information diffusion, communities do not only refer to users who interact with *each other* (usually called cohesive

[2]Relation u, v is considered only if there exists at least one diffusion episode in the training set where u is infected before v.

communities), but also to users who interact with the *same other groups of users* (usually referred to as 2-mode communities). These two types of communities were later studied in [40, 31]. These works tell us two important things about information diffusion :

- If there is diffusion between users a and b in a network, and diffusion between b and c, then user a is more likely to diffuse content to c. This is due to the presence of *cohesive* communities.
- If there is diffusion between a and c, diffusion between b and c, and between a and d, then b is more likely to diffuse content to d. This is due to the presence of *2-mode* communities [40].

In classical cascade models, any sets of probabilities can be defined on links, without any regularization constraints on the extracted diffusion network. Besides leading to overfitting issues, this can lead to unrealistic communication structures (especially when no prior information is given about the social structure of the considered network). On another hand, thanks to geometric constraints on continuous spaces, projecting users in a continuous representation space naturally allows one to capture such social regularities, which cannot be represented in discrete models without additional assumptions about the underlying network. Figure 1 depicts our proposal, which thus consists in moving from a classical influence graph paradigm to an influence continuous space one for the modelization of diffusion dynamics in social networks.

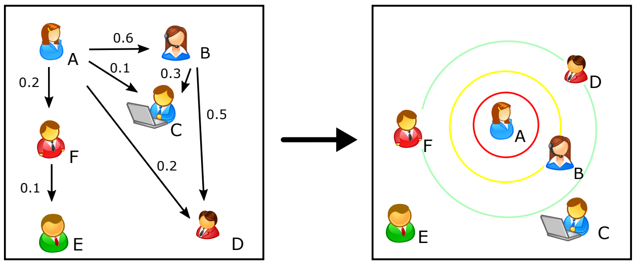

Figure 1: From an influence graph to a continuous influence space. On the left, values associated to edges represent influence probabilities between users. On the right, these influence probabilities are functions of the relative positions of two users. Circles represent equi-probability levels for the diffusion from user A to other users.

Learning representation has been a growing topic recently with applications in many domains [4]. It has been used to learn complex relationships between items like for example in [5]. Since diffusion episodes are essentially temporal series of binary events, learning from sequences is also relevant to the topic of this paper. Representation learning has been used in a series of applications involving sequences or more generally dynamic data. A probabilistic model has been recently proposed in [9] for playlist prediction, where the use of a latent space prevents from having to learn transition probabilities for every single possible pair of songs. While close, our problem differs from this work on the fact that diffusion cascades spread as tree structures, not straight sequences, with possibly simultaneous infections. In our case, influence causalities are hidden: any of the previously infected users can be the source of a given observed new infection. Recently, the idea to use embeddings to predict sequences of elements was also developed in language models such as $Word2Vec$ [26]. Here, embeddings reflect the syntactic and semantic structures of the data sequences (e.g. sentences). Once again, using an embedding is a way to add a regularization over words representations : two words with similar meanings tend to be projected close to each other, and will thus be predicted in a similar context. Recurrent neural networks, which build fixed size embeddings from sequences, have also been used with success in applications like speech recognition [17] or more recently for translation [11].

At last, user embeddings for diffusion tasks have been employed in [6], where infected users are projected near the infection source, the distance reflecting their respective rank of infection. However the goal was different. This model did not attempted to model the diffusion dynamics, but only to determine final infections knowing a unique source of diffusion.

3. EMBEDDINGS FOR DIFFUSION

3.1 Notations

Traditionally, information diffusion on networks is observed as a set $\mathcal{D}$ of "diffusion episodes" $\mathcal{D} = (D_1, D_2, \dots, D_n)$. A diffusion episode D is a set of timestamped user actions related to the same piece of information. It can correspond, for instance, to a set of users who "liked" a specific YouTube video, posted a given url, replied to a given message, etc... It describes to whom and *when* an item spreads through the network, but not *how* diffusion happens: Contrary to cascade structures, the information of who infected who is unknown in such observed inputs. Cascade models usually perform assumptions on this latent factor to build their influence graphs [34].

Given a social network composed of a set of N users[3] $\mathcal{U} = (u_1, ..., u_N)$, a diffusion episode D is then defined as a set of infected users associated with their timestamps of infection: $D = \{(u, t^D(u))|u \in \mathcal{U} \wedge t^D(u) < \infty\}$, where $t^D : \mathcal{U} \rightarrow \mathbb{R}^+$ gives infection timestamps for users infected by the diffusion in concern, or ∞ for non infected ones. Timestamps returned by t^D are relative timestamps given the one of the first infected user (i.e., the source of diffusion, for which t^D is 0). In the following, we note $D(t)$ the set of users who have been infected by diffusion D before timestamp t: $D(t) = \{u \in \mathcal{U}|t^D(u) < t\}$. Symmetrically, we note $\bar{D}(t)$ the set of users who have not been infected before t. $D(\infty)$ is the set of all eventually infected users.

3.2 Diffusion Model

As stated above, we build upon the well-known Independent Cascade Model. We propose to embed it in a continuous space to capture regularities between diffusion relationships of the network and to lighten the memory cost of the model. Based on the relative positions of users, the model allows one to simulate diffusion processes that comply to the observed dynamics.

IC follows an iterative process for simulating cascades of diffusion, where, at each iteration, each newly infected user u gets a unique chance to infect other users v of the network

[3]We talk about users throughout the paper, but the results remain valid for any other kinds of nodes.

with some probability $P_{u,v}$. Here, we consider a slightly modified version of IC, as done in [25], where diffusion between users is assumed to follow uniform delay distributions rather than occurring in successive discrete time-steps. In real life, diffusion indeed occur in continuous time, not discrete. Moreover, [23] recently showed that letting time delay considerations aside when learning transmission probabilities from real-world social data allows one to obtain more robust models since not disturbed by usually noisy infection timestamps. Therefore, as advised in this paper, we only consider partial orders of infections rather than exact infection timestamps to build our model of diffusion.

In our model, the probability for a user to be infected by a diffusion process D thus depends on all previously infected users in D. Given a set of potentially influential users $I \subseteq \mathcal{U}$ (i.e. a set of previously infected users), the probability $P(v|I)$ to observe the infection of a user $v \in \mathcal{U} \setminus I$ knowing this set is therefore defined as:

$$P(v|I) = 1 - \prod_{u \in I}(1 - P_{u,v}) \tag{1}$$

where $P_{u,v}$ stands for the probability of transmission from user u to user v. $P(v|I)$ then corresponds to the probability that at least one user from I succeeded in transmitting content to v. While in IC an individual transmission probability $P_{u,v}$ is estimated for every pair of users (u, v), we propose to use a function of the users latent representation $f : \mathbb{R}^d \times \mathbb{R}^d \rightarrow [0, 1]$, where d is the size of the latent space:

$$P_{u,v} = f(z_u, \omega_v) \tag{2}$$

where $z_u \in \mathbb{R}^d$ is the representation of user u as a sender of a content, while $\omega_v \in \mathbb{R}^d$ is the representation of user v as a receiver of the diffused content. Then, two sets of parameters $Z = (z_u)_{u \in \mathcal{U}}$ and $\Omega = (\omega_u)_{u \in \mathcal{U}}$ are defined to embed the users, one used for the "source" users and the other for "target" ones, to allow the representation of oriented diffusion networks (diffusion relationships are not symmetric in most of networks).

Different choices are possible for f. We performed different tests and came out with the following function that has been used throughout all the experiments in this paper. Since the contagion probabilities are independent one from the other, the only formal requirement is that f gives a value in $[0, 1]$, so that it can be interpreted as a probability.

$$f(z_u, \omega_v) = \frac{1}{1 + exp(z_u^{(0)} + \omega_v^{(0)} + \sum\limits_{i=1}^{d-1}(z_u^{(i)} - \omega_v^{(i)})^2)} \tag{3}$$

where $x^{(i)}$ stands for the i-th component of the vector x. It defines a transmission probability that decreases as the distance between the sender u and the receiver v in the latent space increases. The use of a sigmoid function presents the great advantage of returning a real value in $[0, 1]$ which thus can act as a consistent probability that can directly be used as an influence probability $P_{u,v}$ in our model. Moreover, this function is smoothed at its bounds, which allows variations to have a greater impact for some critical levels which correspond to distances between users between which influence is uncertain. Slight variations of extreme (low or high) values must indeed have less impact on the model probabilities since such values usually concern obvious influences (at least after a sufficient number of learning iterations). Note that we add biases $z_u^{(0)}$ and $\omega_v^{(0)}$ that act as priors on the general tendency to infect other users for u and to become infected for v. These parameters allow the model to get more degrees of freedom, by determining specific functions of probability computations according to who are the users in concern: the same distance between embeddings of two users z_u and ω_v does not lead to the same probability according to who is the sender u and who is the receiver v.

3.3 Learning Algorithm

We follow [34], or more exactly [23] for its time-relaxed version, in the derivation of the likelihood. Considering the set of all pairwise transmission probabilities $\mathcal{P} = \{P_{u,v}|(u, v) \in \mathcal{U}^2\}$, the probability of observing a specific diffusion episode is (if we let time considerations aside and focus on partial orders of infection as advised in [23]):

$$P(D) = \prod_{v \in D(\infty)} P_v^D \prod_{v \in \bar{D}(\infty)} (1 - P_v^D) \tag{4}$$

where $P_v^D = P(v|D(t^D(v)))$ is the probability of observing user v knowing previously infected users in D. Also, it is possible to consider the following log-likelihood for all diffusion episodes from the training set $\mathcal{D}$:

$$\mathcal{L}(\mathcal{P}; \mathcal{D}) = \sum_{D \in \mathcal{D}} \big(\sum_{v \in D(\infty)} log(P_v^D) + \sum_{v \in \bar{D}(\infty)} log(1 - P_v^D)\big) \tag{5}$$

Due to the formulation of the probability P_v^D (equation 1), this log-likelihood is very difficult to maximize directly. However, following the learning methodology described in [34], an Expectation-Maximization (EM) algorithm can be defined to solve this optimization problem. This algorithm considers the following expectation function $\mathcal{Q}(\mathcal{P}|\hat{\mathcal{P}})$:

$$\mathcal{Q}(\mathcal{P}|\hat{\mathcal{P}}) = \sum_{D \in \mathcal{D}} \left(\Phi^D(\mathcal{P}|\hat{\mathcal{P}}) + \sum_{v \in \bar{D}(\infty)} \sum_{u \in D(\infty)} log(1 - P_{u,v}) \right) \tag{6}$$

where $\Phi^D(\mathcal{P}|\hat{\mathcal{P}})$ corresponds to the expected value, for a given diffusion episode D, of the first term of the log likelihood function, which stands for the log likelihood w.r.t. infected users only. It is computed with respect to the conditional distribution of success of diffusion between users under the current estimate of probabilities $\hat{\mathcal{P}}$. Knowing that a user v is infected with an estimated probability $\hat{P_v^D}$ (which is computed via formula 1 with estimated transmission probabilities $\hat{P_{u,v}}$), the conditional probability that the diffusion from a given previously infected user u succeeded is given by $\frac{\hat{P_{u,v}}}{\hat{P_v^D}}$. Considering success and fail cases for each possible transmission of content implies the following expectation term:

$$\Phi^D(\mathcal{P}|\hat{\mathcal{P}}) = \sum_{v \in D(\infty)} \sum_{u \in D(t^D(v))} \frac{\hat{P_{u,v}}}{\hat{P_v^D}} \log(P_{u,v}) + (1 - \frac{\hat{P_{u,v}}}{\hat{P_v^D}}) \log(1 - P_{u,v}) \tag{7}$$

Note that, with transmission probabilities depending on relative positions in a continuous space rather than being set independently for each pair of considered users, the likelihood maximization problem detailed above cannot be decomposed into some independent convex optimization subproblems. Our optimization problem is not convex, however, using stochastic gradient procedures, it is possible to define an efficient learning process which converges to effective local maxima when the number of dimensions is well fitted (a number of $d = 25$ dimensions appears as a good trade-off as observed in section 4).

The pseudo-code of the learning process of our model is given by algorithm 1. It corresponds to an extension of the EM algorithm used in [34], adapted to our non parametric model of the transmission probabilities $\mathcal{P} = \{P_{u,v} | (u,v) \in \mathcal{U}^2\}$. By considering $P_{u,v} = f(z_u, \omega_v)$ for each pair of users (u,v) as proposed in equation 2, the aim is to find optimal parameters Z and Ω that maximize the likelihood given by equation 5. We use a stochastic gradient ascent process to learn parameters sets Z^* and Ω^*. At each iteration, the process:

1. Line 8: Samples a diffusion episode D from $\mathcal{D}$;

2. Line 9: Samples a user v to consider from $\bar{D}(1)$ (the initial users of the diffusion episode are not considered since their infection cannot be explained by any previously infected user);

3. Lines 11 to 18: If v is infected in the diffusion episode D, computes the current estimates $\hat{P_v^D}$ and $\hat{P_{u,v}}$ for all users u infected before v in D. This is done by considering formula 1 and 2 with the current values z_u and ω_v.

4. Lines 19 to 29: Uses a gradient ascent step to update the values of Z and Ω. This update step aims at increasing the value of $\mathcal{Q}(\mathcal{P}|\hat{\mathcal{P}})$ (see equation 6). If the user v is infected in D, the update step involves a derivation of $\Phi^D(\mathcal{P}|\hat{\mathcal{P}})$ (lines 22 to 25). Otherwise, it involves a derivation of $log(1 - P_{u,v})$ (lines 25 to 28). The learning rate ϵ used in our experiments is 10^{-4}.

5. Lines 31 to 38: Tests the convergence of the learning every $freq$ iterations. After having computed all the pairwise transmission probabilities $\mathcal{P}$ using embeddings parameters Z and Ω, the algorithm computes the global log-likelihood by using formula 5. If this log-likelihood has not increased since the last computation, the learning process stops and returns current Z and Ω parameters. In our experiments this convergence test is performed every 1000000 iterations.

Note that, due to our sampling process (lines 8 and 9), every pair (u,v) is not associated with episodes from $\mathcal{D}$ in the same proportion as in formula 6. Without any correction, it would induce a biased learning process where pairs occurring in small episodes would be over-considered compared to their weight in formula 6. Therefore, lines 1 and 10 aim at computing a correction for this sampling related bias. According to our sampling process, the probability to consider the set of transmission probabilities to a given user v for a given episode D equals:

$$\frac{1}{|\mathcal{D}| \times |\bar{D}(1)|}$$

Now, we would like each transmission probability occurrence in formula 6 to be considered uniformly, thus following the

Algorithm 1: Stochastic Learning of Embedded IC

```
   Input:
       $\mathcal{U}$: the set of Users;  $\mathcal{D}$: the set of Diffusion Episodes;
       $d$: the number of dimensions of the projection space;
       $\epsilon$: the learning step;
       $freq$: the frequency of stop tests;
   Output:
       $Z = \{\forall u \in \mathcal{U} : z_u \in \mathbb{R}^d\}$ the sender embeddings ;
       $\Omega = \{\forall u \in \mathcal{U} : \omega_u \in \mathbb{R}^d\}$ the receiver embeddings ;
1  $nbProbas \leftarrow \sum_{D \in \mathcal{D}} \sum_{u \in D(\infty)} |\bar{D}(t^D(u)+1)|$;
2  for $u \in \mathcal{U}$ do
3      $z_u \leftarrow$ Random values in $[-1,1]^d$;
4      $\omega_u \leftarrow$ Random values in $[-1,1]^d$;
5  end
6  $oldL \leftarrow -\infty$;   $it \leftarrow 0$;
7  while $true$ do
8      Sample $D \in \mathcal{D}$;
9      Sample $v \in \bar{D}(1)$;
10     $\alpha \leftarrow \frac{|\mathcal{D}| \times |\bar{D}(1)|}{nbProbas}$;
11     if $t^D(v) < \infty$ then
12         $\hat{P_v^D} \leftarrow 1$;
13         for $u \in D(t^D(v))$ do
14             $\hat{P_{u,v}} = f(z_u, \omega_v)$;
15             $\hat{P_v^D} \leftarrow \hat{P_v^D} \times (1 - \hat{P_{u,v}})$;
16         end
17         $\hat{P_v^D} \leftarrow 1 - \hat{P_v^D}$;
18     end
19     for $u \in D(t^D(v))$ do
20         $\xi_u^+ \leftarrow \frac{\partial \log f(z_u, \omega_v)}{\partial z_u}$;
           $\xi_u^- \leftarrow \frac{\partial \log(1 - f(z_u, \omega_v))}{\partial z_u}$;
21         $\xi_v^+ \leftarrow \frac{\partial \log f(z_u, \omega_v)}{\partial \omega_v}$;
           $\xi_v^- \leftarrow \frac{\partial \log(1 - f(z_u, \omega_v))}{\partial \omega_v}$;
22         if $t^D(v) < \infty$ then
23             $z_u \leftarrow z_u + \alpha \times \epsilon \times (\frac{\hat{P_{u,v}}}{\hat{P_v^D}} \xi_u^+ + (1 - \frac{\hat{P_{u,v}}}{\hat{P_v^D}}) \xi_u^-)$;
24             $\omega_v \leftarrow \omega_v + \alpha \times \epsilon \times (\frac{\hat{P_{u,v}}}{\hat{P_v^D}} \xi_v^+ + (1 - \frac{\hat{P_{u,v}}}{\hat{P_v^D}}) \xi_v^-)$;
25         else
26             $z_u \leftarrow z_u + \alpha \times \epsilon \times \xi_u^-$;
27             $\omega_v \leftarrow \omega_v + \alpha \times \epsilon \times \xi_v^-$;
28         end
29     end
30     $it \leftarrow it + 1$;
31     if $it$ $mod$ $freq = 0$ then
32         $\mathcal{P} \leftarrow \{P_{u,v} | (u,v) \in \mathcal{U}^2 \wedge P_{u,v} = f(z_u, \omega_v)\}$;
33         $L \leftarrow$ Computation of the global log-likelihood
           using formula 5 with probabilities in $\mathcal{P}$;
34         if $L \leq oldL$ then
35             return $(Z, \Omega)$;
36         end
37         $oldL \leftarrow L$;
38     end
39 end
```

probability:

$$\frac{1}{\sum\limits_{D\in\mathcal{D}}\sum\limits_{u\in D(\infty)}|\bar{D}(t^D(u)+1)|}$$

whose denominator is computed as *nbProbas* (line 1) in the algorithm. This corresponds to the sum of counts of considered probabilities in each episode D (the total number of considered probabilities in formula 6). Then, to correct the sampling related bias, we compute (line 10):

$$\alpha = \frac{|\mathcal{D}| \times |\ \bar{D}(1)\ |}{\sum\limits_{D\in\mathcal{D}}\sum\limits_{u\in D(\infty)}|\bar{D}(t^D(u)+1)|}$$

Finally α is used in gradient updates (lines 22 to 28) to readjust the importance of the samples in the log-likelihood to maximize.

4. EXPERIMENTS

4.1 Datasets

Various datasets extracted from the Web have been used for our experiments. These datasets can be divided in two categories : User-Item and User-User datasets.

User-Item Datasets

These datasets essentially consist of triplets of the form : *(user,item,timestamp)*. Each triplet indicates that a given user interacted with some information item at a known time, either by liking it, commenting it, clicking it or sharing it. Each item corresponds to a diffused content, which infects every user who interacted with it. The "source" users are the ones who first interacted with it[4].

Digg is a collaborative news portal on which users can post links to *stories* (articles, blog posts, videos...). Other users can then "digg" these stories if they like it. Stories appear on the front page of Digg based on the amount of "diggs" they have received. We used the Digg stream API to collect the *complete* Digg history (every single story posted, all diggs, and all comments) during a one month time window.

Lastfm is a music streaming website. We used the data collected in [8], which is the complete history of users and songs over one year.

Irvine This dataset was presented in [30]. It contains users posts to forums from an online students community at the University of California, Irvine. Each forum is considered to represent an information item.

Twitter We monitored the tweets of a set of about 5000 active Twitter users during 2 weeks, while the 2012 US Presidential Campaign. We then consider each hashtag to represent one information item spreading in the network.

[4]Note that for these datasets, there does not exist any explicit known graph. In such cases, it is therefore particulary useful to get approaches able to extract influence channels without any external information other than the infection data.

User-User Datasets

In these datasets, only user-user interactions are observed. Users post blog or microblog messages. We use the procedure described in [14] to build the data. Each message can contain a link to another message posted by another user. We consider each set of linked messages to be a diffusion episode gathering users who wrote those messages.

ICWSM The International AAAI Conference on Weblogs and Social Media 2009 published a corpus containing 44 millions of blog posts collected over a 1-year period [7]. We consider each blog to be a "user" in the social network and diffusion episodes are composed of sets of posts which are linked to each other by citations: each connected component of the posts graph is an episode.

Memetracker The memetracker dataset described in [24] contains millions of blog posts and news articles. Each website or blog stands as a user, and we use the phrase clusters extracted by [24] to trace the flow of information.

Online Social Networks datasets tend to be very noisy. To limit the effect of noise, we filter users to only keep a fraction of the most active ones. We also limit the learning set to a relatively small number of diffusion episodes. Limiting the number of learning cascades is necessary to keep classical models (IC, CTIC, NetRate) tractable. It also makes the learning data sparser, which is a more realistic setup, and will help us to observe the generalization ability of our approach.

As explained in section 3.2, we do *not* consider social graphs of the networks to learn the models, since they often turn out to be either irrelevant or are simply unknown in real-life applications. All of the tested models that require a graph are therefore learned on the "co-participation" graph: we created link a link (u, v) if and only if there is at least one diffusion episode D that contains both u and v and $t^D(u) < t^D(v)$.

Table 1 gives some statistics about the datasets. This table reports numbers of users and episodes considered in our experiments. It also reports the average length of diffusion episodes, as well as the co-participation graph number of links and density (i.e. $\frac{\text{nbLinks}}{(\text{nbUsers} \times (\text{nbUsers} - 1)}$).

4.2 Baselines

The following state-of-the-art models are considered in our experiments:

- **IC**: the classic independent cascade model our works grounds in. As done for our model, we consider a relaxed version of IC with uniform delays of infection, since it greatly better fits real-world social data and leads to obtain much better results for the tasks set in our experiments. Weights are learned as defined in [34].

- **Netrate**: as *IC*, *Netrate* [14] is a cascade model which defines influence probability distributions on the network to model information propagation. It considers time-dependent distributions instead of static probabilities as in IC or in our model: parametric time-dependent distributions are learned to best fit with

Table 1: Some statistics about our real datasets

Corpus	Users	Links	Density	Train Episodes	Test Episodes	Avg episode size
Irvine	847	74871	0.1	433	49	14.6
Icwsm	2270	4775	0.001	19027	1000	2.22
Memetracker	498	229073	0.9	10000	1000	2.17
Digg	3295	689416	0.06	17000	1000	2.43
Twitter	2841	884832	0.09	10000	1000	20.5
LastFm	986	708159	0.72	10000	1000	7.25

observed infection timestamps. Note that we only report here results obtained with the *exponential* version of the *NetRate* model, as other distribution laws proposed in [14] (i.e., *power* and *rayleigh* laws) lead us to similar results.

- **CTIC**: As defined in [32], *CTIC* is a continuous-time version of the *IC* model. As *NetRate*, it uses exponential distributions to model delays of diffusion between users, but rather than a single parameter for each 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.

We then describe two sets of experiments used to evaluate the model. The first one is the prediction of infected users and the second one is the prediction of diffusion relationships between users.

4.3 Diffusion Prediction

As true communication probabilities are unknown in the case of real-world social networks, our model cannot be assessed by considering a divergence measure between inferred probabilities $\mathcal{P}$ and a ground truth $\mathcal{P}^*$. We therefore propose to consider a related task of diffusion prediction in which we try to retrieve the set of eventually infected users $D(\infty)$, given a set of initiators $D(1)$.

For all models, training is performed on a training set by maximizing a likelihood. This allows to learn the model parameters (diffusion probabilities in IC, probability distribution parameter values in Netrate and CTIC, representations in our model). All the models are generative so that once a model is trained, it can be used for prediction tasks on a test set. The diffusion prediction task however requires the use of Monte-Carlo simulations to estimate infections probabilities knowing source users of infections. Starting from a seed set of users, each model is run iteratively for each simulation, with successive infections occurring according to the learned diffusion laws. Statistics (e.g. performance measures) can then be computed on the test set. Reported results obtained are averages over 1000 simulations for each diffusion episode of the test set.

Models are evaluated by considering various evaluation measures:

- MSE: predicted probabilities are compared with the actual values of infections for each users (0 or 1) using a mean-squared-error measure;
- LogLikelihood: the average log-likelihood $\mathcal{L}(\mathcal{P};\mathcal{D})$ for all diffusion episodes from the testing set, while rescaling all probabilities in $[10^{-5}, 1-10^{-5}]$ to avoid $log(0)$ problems;
- MAP: the classical mean average precision measure computed over rankings of users in decreasing order of their infection probability;
- F1: the classical F1 measure, with precision and recall scores adapted for the evaluation of predicted diffusions:

$$p = \frac{\sum_{u \in \mathcal{D}(\infty)} P_u^{\mathcal{D}}}{\sum_{u \in \mathcal{U}} P_u^{\mathcal{D}}} \qquad r = \frac{\sum_{u \in \mathcal{D}(\infty)} P_u^{\mathcal{D}}}{|\mathcal{D}(\infty)|}$$

$$F1 = \frac{2 * p.r}{p + r}$$

Before comparing our model to state of the art models presented above, we first consider the impact of the number of dimensions of the projection space on the processing time required for the learning step to converge (see algorithm 1 for details about the stop criterion) and the effectiveness of the model to predict test episodes. We report here experiments performed on the Digg dataset, but similar observations can be done on other ones. Figure 2 plots results obtained on a Intel(R) Core(TM) i7 CPU 950@3.07GHz according to different numbers of dimensions of the projection space, on a logarithmic scale. From this, we can note that the processing time of the learning (CPU time given in seconds) increases logarithmically with regards to the number of dimensions, while very few increase of effectiveness (in term of log-likelihood on the test episodes) is observed with numbers of dimensions greater than 25. In the following, we therefore set the number of dimensions of the projection space to $d = 25$, which appears as a good trade-off between processing time and model accuracy.

Table 2 presents average effectiveness results obtained by the different experimented approaches on the various datasets described above, in term of their ability to predict test episodes according to measures of MSE, log-likelihood, MAP and F1 described above. This table highlights results that are significantly better than IC with a exponent star. Best results obtained for a corpus and a given measure are given in bold. The first thing that should be noted is that our model performs *at least as well* as all other experimented models for all datasets. It obtains the best result in almost all cells, except the MAP for Irvine and F1 for Digg for which it obtains a slightly lower score than some other models but without observing a significant difference with them. Relatively low results obtained by CTIC and NetRate can be explained by the fact that, as recently shown in [23], no delay regularities can be observed in the data and that taking them into account disturb the extraction of transmission probabilities. The superior results obtained by our model show that this approach has been able to successfully embed the probabilities $P_{u,v}$ in a latent space. This performance is obtained with a much lower number of parameters

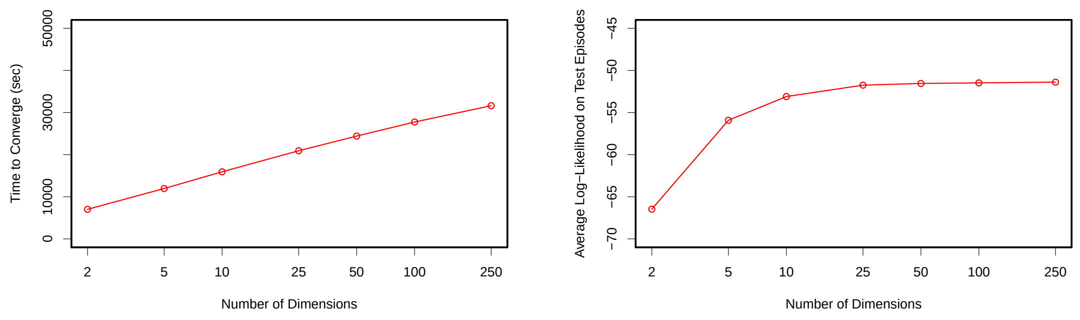

Figure 2: Time (in seconds) for the learning step to converge and average log-likelihood results obtained on test episodes, according to different numbers of dimensions of the projection space, on a logarithmic scale. Results obtained on the Digg dataset.

Corpus	Model	MSE	LogLikelihood	MAP	F1	nbParams
Irvine	IC	15,31	-960,5	0,079	0,020	74871
	NetRate	15,42	-892,13	0,078	0,019	74871
	CTIC	15,29	-771,42*	**0,080**	0,020	149742
	Embedded IC	**14,53***	**-532,5***	0,079	**0,025***	42350
ICWSM	IC	0,2	-8,3	0,77	**0,651**	4775
	NetRate	0,23	-9,01	0,72	0,357	4775
	CTIC	0,22	-8,46	0,76	0,482	9550
	Embedded IC	**0,19**	**-6,14***	**0,78**	**0,651**	113500
MemeTracker	IC	32,62	-795,85	0,22	0,0585	229073
	NetRate	34,55	-850,48	0,17	0,0442	229073
	CTIC	33,27	-802,52	0,22	0,0551	458146
	Embedded IC	**32,15**	**-791,3**	**0,23**	**0,0632***	24900
Digg	IC	2,1	-69,5	0,411	**0,201**	689416
	NetRate	1,95	-64,01*	0,409	0,199	689416
	CTIC	1,92*	-64,18*	0,413	**0,201**	1378832
	Embedded IC	**1,79***	**-51,75***	**0,434***	0,198	164750
Twitter	IC	6,70	-412,75	0,047	0,012	884832
	NetRate	6,91	-428,78	0,039	0,011	884832
	CTIC	6,72	-401,56	0,049	0,012	1769664
	Embedded IC	**5,47***	**-223,15***	**0,056***	**0,013**	142050
LastFM	IC	12,13	-409,5	0,132	0,026	708159
	NetRate	13,91	-413,02	0,112	0,022	708159
	CTIC	12,12	-409,3	0,128	0,025	1416318
	Embedded IC	**11,62***	**-405**	**0,151***	**0,027**	49300

Table 2: Results obtained on real-world datasets. Values with a star are significantly better than IC ones (95% student t-test). Values in bold correspond to the best results obtained for the experiment in concern.

than for all other approaches (except on Icwsm, which owns a very low density as observed in table 1), as shown in the rightmost column. We can see that all measures behave differently. This is linked to datasets properties:

- The MSE of Embedded IC is significantly better on the User-items datasets: Irvine, Digg, and LastFM. On these datasets, diffusion is actually difficult to detect : one user's liking the same item as another one does not necessarily means that one *influenced* the other. These two users may just happen to have similar centers of interest [1]. It is difficult to learn a IC model in this context. Thanks to its latent space, our Embedded IC is able to identify diffusion links between users, which leads to a better MSE. See subsection 4.4 for additional results about this property of Embedded IC.

- The Log-Likelihood of our model is better on the Irvine, Icwsm, Digg and Twitter datasets. These datasets happen to have something in common: they have a much lower co-participation graph density (table 1). This means that the IC model can only learn a very small amount of transmission probabilities, and is unable to predict diffusion between the majority of users pairs. Our Embedded IC, however, can infer diffusion probabilities for *all* user pairs.

- Finally, Embedded IC achieves a higher MAP on the Digg, Twitter and LastFM datasets. These websites also have a specific property in common: they behave more like "information portals". When a user arrives on Digg or LastFM, the first thing he finds is a list of popular items, a list of new items, and some recommendations based on his browsing history. User-to-User interactions are much less important. On Twitter, the User-to-User interactions are very important, but there also is a "Trending Topics" sections. This means that on those websites, diffusion happens on a more "global" scale: the more popular an information is, the more likely it is to infect new users. This phenomenon is better captured by our embedded approach, where each user has at least a small chance to infect *any* other one.

Overall, these results highlight various advantages of our approach over IC: less parameters to learn, better generalization and the ability to model different types diffusion. But they also show us that diffusion remains a complex phenomenon, that can take a lot of different forms. All the datasets we collected are quite different in term of size and density. On top of that, each one has its own mechanism of diffusion (retweets, messages threads, word-of-mouth...). This explains why all the models behave differently on each dataset and why we needed several measures to captures several properties. The problem of Information Diffusion Prediction is not only hard, it also appears to be very specific to the website to which one wishes to apply it.

4.4 Influence Relationships Detection

For a second evaluation, we applied the models for a task of detection of interactions from diffusion episodes. This allows us to better analyze the ability of our approach to identify important diffusion channels between users. Given a population and a set of diffusion episodes over this population, the goal is to find which influence relationships are the most likely to exist between users. This task has been extensively studied (see [14, 15] and references therein).

We used the Memetracker dataset for this experiment, which was used for a similar task by [14]. We extract an actual ground truth set of links E: for each post of website v containing a hyperlink to website u, we put a link (u, v) in E. It is important to note that the existence of a link (u, v) means that we *actually observed* a specific piece of information going from u to v.

For each pair of users (u, v), the learned probability of transmission $P_{u,v}$ is interpreted as the probability for a link to exist between them. We sort the probabilities obtained by the models in descending order, then plot a Precision-Recall curve to evaluate results.

Results are presented in figure 3 as Precision/ Recall curves. It can be observed that our Embedded IC model clearly outperforms all the other models on this task. This illustrates the ability of our approach to infer links that are not present in the training set and to identify important ones. Discrete cascade models will always learn a transmission rate of 0 on links (u, v) such that no diffusion episode in the training set contains user u then user v. Such relationship may however exist in the actual graph, and our model is able to retrieve some of them. Most importantly, like we said above, ground truth links are the ones on which we observed actual information diffusion. We can therefore be sure that those links are influence links, and not just homophily relations [1].

5. CONCLUSION

In this paper, we have presented an embedded version of the Independent Cascade Model to predict information diffusion in Online Social Networks. Based on observations of the community structures in social network, we proposed to project users of a social network in a latent space, and then

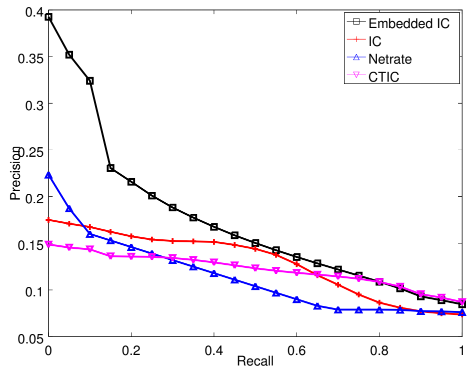

Figure 3: Recall-Precision of link prediction on the Memetracker Dataset.

to use the distances between them to compute the probabilities of transmission.

We have performed experiments on a large collection of datasets. Results show that we are able to correctly embed users to reach usually better, else at least the same, performance levels as IC, while learning less parameters.

Future works will focus on several possibilities of extension. The most obvious one would be to use the *type* or the *content* of diffusing information to refine our prediction. In our setting, this comes down to learn a content-dependent metric between users representations in our projection space. Finally, the stochastic algorithm we presented could be adapted to an online context, to give a "live" prediction of diffusion as infections are observed.

6. ACKNOWLEDGMENTS

This work has been partially supported by the REQUEST project (projet Investissement d'avenir, 2014-2017), the LOCUST project (ANR, 2015-2018) and the project ARESOS from the CNRS program MASTODONS.

7. REFERENCES

[1] A. Anagnostopoulos, R. Kumar, and M. Mahdian. Influence and correlation in social networks. In *KDD'08*, pages 7–15. ACM, 2008.

[2] E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic. The role of social networks in information diffusion. In *WWW'12*, pages 519–528. ACM, 2012.

[3] N. Barbieri, F. Bonchi, and G. Manco. Cascade-based community detection. In *WSDM '13*, pages 33–42, New York, NY, USA, 2013. ACM.

[4] Y. Bengio, A. C. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013.

[5] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS'13*, pages 2787–2795, 2013.

[6] S. Bourigault, C. Lagnier, S. Lamprier, L. Denoyer, and P. Gallinari. Learning social network embeddings for predicting information diffusion. In *WSDM '14*, pages 393–402, New York, NY, USA, 2014. ACM.

[7] K. Burton, A. Java, and I. Soboroff. The icwsm 2009 spinn3r dataset. In *ICWSM'09*, May 2009.

[8] O. Celma. *Music Recommendation and Discovery in the Long Tail*. Springer, 2010.

[9] S. Chen, J. L. Moore, D. Turnbull, and T. Joachims. Playlist prediction via metric embedding. In *KDD'12*, pages 714–722. ACM, 2012.

[10] J. Cheng, L. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec. Can cascades be predicted? In *WWW'14*, pages 925–936, New York, NY, USA, 2014. ACM.

[11] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP'14*, pages 1724–1734, 2014.

[12] N. Du, L. Song, M. Yuan, and A. J. Smola. Learning networks of heterogeneous influence. In *NIPS'12*, pages 2780–2788. Curran Associates, Inc., 2012.

[13] J. Goldenberg, B. Libai, and E. Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing letters*, 12(3):211–223, 2001.

[14] M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *ICML-11*, pages 561–568. ACM, 2011.

[15] M. Gomez Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. In *KDD '10*, New York, NY, USA, 2010. ACM.

[16] M. Granovetter. Threshold Models of Collective Behavior. *American Journal of Sociology*, 83(6):1420–1143, 1978.

[17] A. Graves, A. Mohamed, and G. E. Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP'13*, pages 6645–6649, 2013.

[18] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins. Information diffusion through blogspace. In *WWW '04*, pages 491–501, New York, NY, USA, 2004. ACM.

[19] A. Guille and H. Hacid. A predictive model for the temporal dynamics of information diffusion in online social networks. In *WWW '12 Companion*. ACM, 2012.

[20] A. Guille, H. Hacid, C. Favre, and D. A. Zighed. Information diffusion in online social networks: A survey. *SIGMOD Rec.*, 42(2):17–28, July 2013.

[21] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD '03*, pages 137–146. ACM, 2003.

[22] C. Lagnier, L. Denoyer, E. Gaussier, and P. Gallinari. Predicting information diffusion in social networks using content and user's profiles. In *ECIR '13*, 2013.

[23] S. Lamprier, S. Bourigault, and P. Gallinari. Extracting diffusion channels from real-world social data: a delay-agnostic learning of transmission probabilities. In *ASONAM'15*, pages 178–185. IEEE Computer Society, 2015.

[24] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD '09*, pages 497–506, NY, USA, 2009. ACM.

[25] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen. Sparsification of influence networks. In *KDD '11*, pages 529–537, NY, USA, 2011. ACM.

[26] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS'13*, pages 3111–3119. Curran Associates, Inc., 2013.

[27] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *IMC '07*, pages 29–42, New York, NY, USA, 2007. ACM.

[28] A. Najar, L. Denoyer, and P. Gallinari. Predicting information diffusion on social networks with partial knowledge. In *WWW '12 Companion*, pages 1197–1204, New York, NY, USA, 2012. ACM.

[29] P. Netrapalli and S. Sanghavi. Learning the graph of epidemic cascades. *SIGMETRICS Perform. Eval. Rev.*, 40(1):211–222, June 2012.

[30] T. Opsahl and P. Panzarasa. Clustering in weighted networks. *Social networks*, 31(2):155–163, 2009.

[31] M. Sachan, A. Dubey, S. Srivastava, E. P. Xing, and E. Hovy. Spatial compactness meets topical consistency: Jointly modeling links and content for community detection. In *WSDM '14*, pages 503–512, New York, NY, USA, 2014. ACM.

[32] K. Saito, M. Kimura, K. Ohara, and H. Motoda. Learning continuous-time information diffusion model for social behavioral data analysis. In *ACML '09*, pages 322–337, Berlin, Heidelberg, 2009. Springer-Verlag.

[33] K. Saito, M. Kimura, K. Ohara, and H. Motoda. Generative models of information diffusion with asynchronous timedelay. *Journal of Machine Learning Research - Proceedings Track*, 13:193–208, 2010.

[34] K. Saito, R. Nakano, and M. Kimura. Prediction of information diffusion probabilities for independent cascade model. In *KES '08*, pages 67–75. Springer-Verlag, 2008.

[35] K. Saito, K. Ohara, Y. Yamagishi, M. Kimura, and H. Motoda. Learning diffusion probability based on node attributes in social networks. In M. Kryszkiewicz, H. Rybinski, A. Skowron, and Z. W. Ras, editors, *ISMIS*, volume 6804 of *LNCS*, pages 153–162. Springer, 2011.

[36] H. Su, A. Gionis, and J. Rousu. Structured prediction of network response. In *ICML'14*, pages 442–450. JMLR Workshop and Conference Proceedings, 2014.

[37] G. Ver Steeg and A. Galstyan. Information-theoretic measures of influence based on content dynamics. In *WSDM'13*, pages 3–12, New York, NY, USA, 2013. ACM.

[38] L. Wang, S. Ermon, and J. E. Hopcroft. Feature-enhanced probabilistic models for diffusion network inference. In *ECML PKDD'12*, pages 499–514. Springer-Verlag, 2012.

[39] J. Yang and J. Leskovec. Modeling information diffusion in implicit networks. In ICDM'10, pages 599–608, Washington, DC, USA, 2010. IEEE Computer Society.

[40] J. Yang, J. McAuley, and J. Leskovec. Detecting cohesive and 2-mode communities indirected and undirected networks. In *WSDM '14*, pages 323–332, New York, NY, USA, 2014. ACM.