Learning Stochastic Models of Information Flow

Luke Dickens #1, Ian Molloy *2, Jorge Lobo *3, Pau-Chen Cheng *4, and Alessandra Russo #5

Abstract—An understanding of information flow has many applications, including for maximising marketing impact on social media, limiting malware propagation, and managing undesired disclosure of sensitive information. This paper presents scalable methods for both learning models of information flow in networks from data, based on the Independent Cascade Model; and predicting probabilities of unseen flow from these models. Our approach is based on a principled probabilistic construction and results compare favourably with existing methods in terms of accuracy of prediction and scalable evaluation, and we describe how to evaluate a broader range of queries than previously shown. Exact evaluation of flow probabilities is exponential in the number of edges and naive sampling can also be expensive, so we propose sampling in an efficient Markov-Chain Monte-Carlo fashion using the Metropolis-Hastings algorithm, described in detail in the paper. Other models make simplifying assumptions about information propagation, for example, constraining the network topology or treating information as a fluid; our model is flexible without requiring any such constraints, and can be used to learn joint and conditional flow probabilities, as well as distributions over flow likelihood. We identify two types of data, those where the paths of past flows are known – attributed data, and those where only the endpoints are known – unattributed data. Both data types are addressed in this paper, including training methods, example real world data sets, and experimental evaluation. In particular, we investigate flow data from the Twitter micro-blogging service, exploring the flow of messages through retweets (tweet forwards) for the attributed case, and the propagation of hashtags (metadata tags) and urls for the unattributed case.

I. INTRODUCTION

We live in an increasingly connected world, in which information is a first order currency. Here, meta-information – information about information – takes on a value of its own, and one flavour of this commodity concerns how information behaves when it resides on a network of repository nodes, and propagates across edges between these nodes. Models of information flow through a network can be used to exploit the communication potential of social networks, to control sharing of sensitive information within organisations, and to predict malware propagation, among other uses. This paper presents efficient methods to predict information flow in networks, based on the Independent Cascade Model, which compare favourably with existing methods in terms of accuracy of prediction and scalable evaluation, as well as supporting a broader range of queries. Hence, this work may be of interest to those concerned with: public health announcements; marketing and advertising; malware propagation; reputation management; and assessing or limiting the damage associated with the undesired disclosure of sensitive information.

Information networks can be arbitrarily complex structures, and the flow dynamics can be sensitive to the (potentially asymmetric) relationships between each pair of nodes. Small changes in the network, such as the addition or removal of nodes or edges, can significantly alter the aggregate behaviour of the network. Ideally, a model of such a system would accurately predict a wide range of properties, including node-to-node flow probability, conditional flow, flow dependent path likelihood, and source-to-community flow. These information networks can have millions (or more) of nodes and may be dynamic, gaining and losing nodes and edges all the time. So, robust models should be able to absorb network changes efficiently, and extrapolate new behaviour when these changes are incorporated. Perhaps most challengingly, many application areas are concerned with stability as well as mean behaviour of systems, so we may need to go beyond predicting expected flows, to estimating probability distributions over flow, e.g., for risk-aware applications.

In this paper, we present a model of information flow through a network, as a directed graph where the nodes are information repositories and the edges are the routes by which information may be shared, based on the Independent Cascade Model of information flow in networks [1], [2], [3], [4]. For example, if this models a social network of users, then the individual nodes may represent the users themselves, and the edges would then be their social relationships. Information in our model refers to the content, rather than an anonymous collection of bits. Other models make simplifying assumptions about information propagation, for example, constraining the network topology to be a directed acyclic graph, e.g., [5] or treating information as having fluid-like properties [6]; our approach instead treats information as propagating atomic objects on general directed graphs. Thus, an atom of information can only ever traverse each edge, and only arrives at each node, a maximum of once; this edge, and its child node, is then described as active for that information. Once active for a given information atom, an edge or node cannot become inactive, and this reflects the fact that, once stored at a node, there is no increase in information content at that node if...
that atom arrives again. For a given network, we assume that if it is possible for a piece of information to traverse an edge, then it will do so with an activation probability (sometimes called a diffusion probability [3]) that depends only on that edge. Consequently, our model estimates the aggregate activation probability of an edge for an anonymous information packet; prediction accuracy may be improved by considering content, time delay, and prior network flow behaviour of each information packet, and we intend to address these concerns in future work.

In general, the activation probabilities in a network will not be known a priori, and instead must be learnt from available evidence. We divide evidence into two types, attributed and unattributed, and show methods for approximating activation probabilities from both types. Learning from attributed evidence is more straightforward, although this may require preprocessing of the data to establish the evidence. Here the learnt model approximates activation probabilities with a beta distribution at every edge, relating to the frequency of activation of that edge in the training set. Unattributed data presents a greater challenge, and we give a new method for learning from this evidence, which improves upon the current state of the art in terms of efficient use of the data. As with the attributed case, we are able to capture the uncertainty in the data by storing distributions over activation probabilities, rather than the point probability estimates learnt in previous work [2], [4]. This allows us to make more efficient use of evidence, and hence predict activation and flow probabilities more accurately, as we demonstrate in experiment.

The exact evaluation of node-to-node flow probabilities in the model is exponential in the number of edges, and naive sampling can also be expensive. We propose a method for sampling an approximation efficiently (linearithmic) using the Metropolis-Hastings algorithm. This method is highly scalable, and can be used to learn joint and conditional flow probabilities, which might not be possible if we were to base our evaluation procedure on a graph walking method for evaluating similarity metrics, such as random walk with restart [7], [8]. We illustrate the soundness of both attributed and unattributed approaches on synthetic graphs, and real data based on Twitter messages, using retweeted messages for attributed data and the mentioning of hashtags and urls for unattributed data. This includes an assessment of the evaluation of conditional and joint probabilities, and how well the models capture uncertainty.

To summarize, our key contributions are:

- Provide a model to learn edge probabilities with uncertainty from unambiguous attributed data, typical where the social graph is known, such as FaceBook or Google+.
- A model to learn edge probabilities with uncertainty from ambiguous, unattributed data, typical of ad hoc channels where citations are uncommon, such as blogs or email.
- An efficient method to learn joint and conditional information flows, as well as dispersion, using the Metropolis-Hastings algorithm.
- An illustration of the soundness of our approaches on real-world data drawn from Twitter.
- Comparison with alternative approaches to learn information flow networks [2], [4].
- A moderate change to the expectation maximization steps of [4] that may significantly improve performance.

The paper is structured as follows. The next section details our information flow model, shows that exact evaluation is highly intractable and describes our method to approximate models from attributed evidence. A more efficient evaluation method using Metropolis-Hastings sampling is next demonstrated. Experiments on attributed data follow, both on synthetically generated data, and on data drawn from the Twitter social networking site. This includes a description of how reTweet messages from Twitter were processed to produce attributed evidence. The unattributed data learning method is then described. A second round of experiments evaluates the accuracy with which synthetic graphs can be modeled from their data, the predictive power of these models on real data, and a comparison with the state of the art. The paper ends with a more detailed discussion of the applications of our approach and future work.

II. INFORMATION FLOW MODEL

This section describes the independent cascade model (ICM), detailing the mechanism of propagation and techniques for approximating ICMs from evidence. Here, an ICM is a directed graph \( G = (V, E, P) \), where \( V \) and \( E \subset V \times V \) are the \( n \) vertices and \( m \) edges respectively. \( P : E \rightarrow [0, 1] \), maps each edge to its associated activation probability.

An ICM represents the stochastic properties of any information object, \( i \), a copy of which potentially resides at each vertex. If \( i \) is at \( v \), then \( v \) is active for \( i \) (i-active). For information object \( i \), there is a set of source vertices, \( V^i_0 \subset V \); each member of \( V^i_0 \) is i-active. For any i-active node \( v_j \), an outgoing edge \( e_{j,k} \) (to node \( v_k \)) is also i-active with probability \( p_{j,k} = P(e_{j,k}) \), meaning that \( i \) traverses \( e_{j,k} \); \( e_{j,k} \) is otherwise inactive for \( i \) (i-inactive). Any vertex \( v_j \notin V^i_1 \) with one or more i-active incoming edges, is also i-active. Any vertex \( v_j \notin V^i_0 \) with no i-active incoming edges, is i-inactive. The set of vertices and edges that are i-active are denoted \( V^i_1 \) and \( E^i \) respectively. Clearly, \( V^i_0 \subseteq V_i \).

A description of all i-active nodes and edges, along with all i-inactive edges with i-active parents, is the active-state of \( i \). A pseudo-state assigns each edge to be i-active or i-inactive, irrespective of the i-activity of the parent node. Computationally pseudo-states are easier to work with than active-states. Given \( i \)'s source nodes \( V^i_0 \) and a pseudo-state, we can derive the active state for \( i \). We discuss active-states and pseudo-states formally and in more detail in section III.

For a vertex \( v_k \in V_i \setminus V^i_0 \), there is a flow of \( i \) from \( V^i_0 \) to \( v_k \). This means \( i \) has made its way to \( v_k \) through a sequence of edges from some \( v_j \in V^i_0 \) to \( v_k \); here \( v_k \) is the sink node. If there is only one member of \( V^i_0 \), i.e. \( \{v_j\} = V^i_0 \), and \( v_k \in V_i \) then we say there is an (end-to-end) flow of \( i \) between
v_j and v_k, written (v_j \sim v_k); for no flow we write (v_j / \sim v_k) - the subscript is dropped when unambiguous. We also consider flow to multiple sink nodes, called source-to-community flow. Flow probability conditioned on the existence (or non-existence) of other flow, is conditional flow.

Experiments in this paper use data from the social networking site Twitter to examine how messages (originating with individual users) are retweeted (analogous to email forwarding), and thus percolate through the social network. We model this as an information flow network, where a message is an information object, users' accounts are the nodes, a node is i-active with respect to a message i if the nodes' user either tweeted i (the source of the tweet) or retweeted i. We assume that the probability of a retweet is a characteristic of the relationship between the previous tweeter and the retweeter, and is the activation probability of the edge between the associated nodes.

An ICM can be used to evaluate flows exactly. To see how, first consider the acyclic graph with three nodes, v_1, v_2, v_3, three arcs, (v_1, v_2), (v_1, v_3), (v_2, v_3), and information object i, where \{v_i\} = Y_i^e. For some flow v_1 \sim v_3, there must be direct flow either from v_1 to v_3, written v_1 \rightarrow v_3; the two direct flows v_1 \rightarrow v_2 and v_2 \rightarrow v_3, written v_1 \rightarrow v_2 \rightarrow v_3; or both. Therefore

\[
\Pr[v_1 \sim v_3] = 1 - (1 - \Pr[v_1 \rightarrow v_2]) (1 - \Pr[v_1 \rightarrow v_3])
\]

(1)

where \( p_{j,k} \) is given by \( P(e_{j,k}) \) as discussed.

The flow probability \( v_1 \sim v_3 \) can be rewritten in terms of the activation probabilities on \( v_3 \)'s incident arcs and the overall flow to the parents of these arcs, i.e.,

\[
\Pr[v_1 \sim v_3] = 1 - (1 - \Pr[v_1 \sim v_2]) (1 - \Pr[v_1 \sim v_1]) p_{1,3}
\]

where \( \Pr[v_j \sim v_j] = 1 \) trivially.

If we add the arc \( (v_3, v_2) \) to form a cycle in the graph, the flow probability \( \Pr[v_1 \sim v_3] \) is also as given in Equation (1). However, the same rewriting is not possible here, because here the flow probability \( \Pr[v_1 \sim v_2] \) depends on a flow path through \( v_3 \), i.e., \( v_1 \rightarrow v_3 \rightarrow v_2 \). To generalise the above rewriting process to support cyclic graphs, we must consider node-to-node flow that excludes other nodes. To indicate a flow from some node \( v_j \) to another \( v_k \) that excludes a set of nodes \( \mathcal{X} \), we write \( (v_j \sim v_k) \text{ ex. } \mathcal{X} \). Now we can rewrite the probability of flow \( v_1 \sim v_3 \) in our cyclic graph as

\[
\Pr[v_1 \sim v_3] = 1 - (1 - \Pr[v_1 \sim v_2] p_{2,3}) (1 - \Pr[v_1 \sim v_1] p_{1,3})
\]

Equation (2) indicates how any end-to-end flow probability can be evaluated recursively for any ICM within \( O(n^2) \sim O(n^{2n}) \) time. This paper uses approximate solutions, found by Metropolis-Hastings sampling, at significantly shorter timescales.

A. Training from Attributed Evidence

For a real information flow network, we cannot know the activation probabilities directly, and must approximate them from evidence. Most generally, evidence sets consist of a collection of information objects each with a set of known flows, i.e., \{\{u, v\} \mid (u \sim v)\} for each i. We describe these evidence sets as unattributed, because when a node becomes i-active for some i, it is not known which incident node in the graph was a direct cause of that activity. Learning from unattributed evidence is conceptually more challenging, can involve joint distributions over activation probabilities, and is addressed in Section V. Conversely, some data-sets contain sufficient information, such that for each non-source i-active node we can directly attribute an incident node as cause of the activity, and we describe evidence with this information as attributed. Formally, attributed evidence is a tuple of objects and their attributed flow, \( D = (\mathcal{O}, \mathcal{F}) \), where the attributed flow is a set of sources, active nodes, and active edges for each object, so \( \mathcal{F} = \{\{\mathcal{V}_i^o, \mathcal{V}_i, \mathcal{E}_i\} \mid i \in \mathcal{O}\} \). Some preprocessing work may be required to extract attributed evidence from raw data, and in Section IV we describe how we achieved this with data from the micro-blogging service Twitter. For attributed evidence, we can establish an independent beta distribution for each activation probability, representing a sequence of bernoulli trials on the evidence set. These approximate ICMs are called betaICMs, and represent probability distributions over possible point probability ICMs.

Thus, a betaICM is a graph \( G = (\mathcal{V}, \mathcal{E}, \mathcal{B}) \), where \( \mathcal{V} \) and \( \mathcal{E} \) are as they were for the exact ICM, and \( \mathcal{B} : \mathcal{E} \rightarrow [1, \infty]^2 \) maps each edge to its associated beta distribution, where for edge \( e_{j,k} \in \mathcal{E} \), we have \( \mathcal{B}(e_{j,k}) = (\alpha_{j,k}, \beta_{j,k}) \) - the \( \alpha \beta \)-parameters of the beta distribution associated with \( e_{j,k} \). Each beta distribution represents our best available knowledge about the activation probability of the associated edges, as based on the evidence set. A betaICM is trained, from evidence, as follows:

1) Set all values \( \alpha_{j,k}, \beta_{j,k} = 1 \).
2) For each object \( i \in \mathcal{O} \) and for each edge \( e_{j,k} \in \mathcal{E} \):
   a) If \( e_{j,k} \in \mathcal{E}_i \), increment \( \alpha_{j,k} \) by one.
   b) If \( v_j \in \mathcal{V}_i \) but \( e_{j,k} \not\in \mathcal{E}_i \), increment \( \beta_{j,k} \) by one.
3) Return all \( \alpha_{j,k} \) and \( \beta_{j,k} \).

We show in Equation (2) how to evaluate end-to-end flow in a point-probability ICM, i.e., with known activation probabilities. The same method can be applied in a betaicm, if we first transform it into the expected point-probability ICM, where the activation probability for each edge, \( e_{j,k} \), as \( p_{i,j} = \frac{\alpha_{i,j}}{\alpha_{i,j} + \beta_{i,j}} \). The next section shows how the Metropolis-Hastings algorithm can be used to efficiently sample end-to-end, source-to-community and conditional flow probabilities from point-probability ICMs and directly from betaicms.
III. Metropolis Hastings Approximation

Metropolis-Hastings sampling is an instance of Markov-Chain Monte Carlo sampling. These Markov-Chain approaches do not produce sequences of random variables that are independently distributed, but can produce sets of random variables that are distributed appropriately. These sample sets can in turn be used to approximate expectations over the random variable.

A. Sampling of Information Flow

This section defines the pseudo-state, demonstrates that a set of flows is unambiguously determined by an underlying pseudo-state, and shows how to predict flow probabilities by sampling pseudo-states. For some point probability model \( \mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{P}) \) with \( n \) nodes and \( m \) edges, a pseudo-state partitions the set of edges into active or inactive, and can be represented as a vector of booleans \( x \in \mathcal{X} \equiv \{0,1\}^m \). We label elements of \( x \in \mathcal{X} \) with the indices of the corresponding edges, so \( x_{j,k} \) is the activity of edge \( e_{j,k} \) in \( x \). Therefore,

\[
\Pr[x \mid \mathcal{M}] = \prod_{x_{j,k} \in \mathcal{E}} p_{j,k}^{x_{j,k}}(1 - p_{j,k})^{(1-x_{j,k})}
\]

(3)

This is costly to sample directly, and in Section III-B we show how Metropolis-Hastings sampling makes this more efficient.

An active-state, \( s \), only need specify the activity of those edges with active parent nodes, and any pseudo-state that assigns the same activity to the same edges is said to give rise to \( s \). We write, \( x \succ \succ s \), if pseudo-state \( x \) gives rise to active-state \( s \), and \( s = \{x \in \mathcal{X} \mid x \succ \succ s\} \in \mathcal{P}(\mathcal{X}) \). Clearly,

\[
\Pr[s \mid \mathcal{M}] = \sum_{x \succ \succ s} \Pr[x \mid \mathcal{M}]
\]

(4)

So to sample active-states, we can sample pseudo-states and derive the active-state.

Each active-state, in turn, gives rise to a set of end-to-end flows (a community of flows), and we write \( s \succ (u \sim v) \) if the flow \( u \sim v \) is present in \( s \), and likewise \( x \succ (u \sim v) \) if \( x \succ s \) and \( s \succ (u \sim v) \) for some \( s \). So equally,

\[
\Pr[u \sim v \mid \mathcal{M}] = \sum_{x \succ (u \sim v)} \Pr[x \mid \mathcal{M}] \approx \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} I(u, v; x)
\]

(5)

where \( \mathcal{D} \) is a large set of pseudo-states sampled from \( \mathcal{M} \), and indicator \( I(u, v; x) = 1 \) if \( x \succ (u \sim v) \), and 0 otherwise.

To evaluate conditional flow probabilities, we need to evaluate the probability of pseudo-state, given certain flow conditions. Conditions are sets of constrained flow, where each constrained flow is a tuple \((u, v, a) \in \mathcal{V} \times \mathcal{V} \times \mathcal{B}\), such that \( a = 1 \) enforces \( u \sim v \) and \( a = 0 \) enforces \( u \not\sim v \). For flow conditions \( C \in \mathcal{P}(\mathcal{V} \times \mathcal{V} \times \mathcal{B}) \), we can use Bayes rule to give

\[
\Pr[x \mid \mathcal{M}, C] = \frac{\Pr[x, C \mid \mathcal{M}]}{\Pr[C \mid \mathcal{M}]}
\]

(6)

We next show how we can use the Metropolis-Hasting algorithm to sample from the marginal distribution (Equation (3)), or conditional flow (Equation (6)).

B. Metropolis Hastings for Information Flow

In the Metropolis-Hastings approach we generate a Markov-Chain of pseudo-states, for which a sufficiently large sequence is representative of the distribution induced by the model \( \mathcal{M} \) and flow conditions \( C \). By conditionally accepting or rejecting some states, Metropolis-Hastings ensures the Markov-Chain converges to the distribution we are sampling from. Metropolis-Hastings is a well established sampling method [9], [10], and we focus on applying this to ICMs.

This uses a generating transition function \( q(\cdot \mid \cdot, \mathcal{M}) : \mathcal{X} \times \mathcal{X} \to [0,1] \), and a state probability function, \( p(\cdot \mid \mathcal{M}, C) : \mathcal{X} \to [0,1] \). For a current state \( x_t \), \( q \) generates a candidate next state, \( x' \), with probability, \( q(x' \mid x_t, \mathcal{M}) \), is unaffected by flow conditions, and represents a first order Markov-Chain, \( p(x \mid \mathcal{M}, C) \) is the probability of pseudo-state \( x \) given the model and conditions. We do not have to evaluate \( p(x_t \mid \mathcal{M}, C) \) fully, it is sufficient, given a current state, \( x_t \), and candidate next state, \( x' \), to evaluate the ratio of the state probabilities

\[
q_{ratio}(x_t, x' ; \mathcal{M}) = \frac{q(x' \mid x_t, \mathcal{M})}{q(x_t \mid x', \mathcal{M})}
\]

(7)

Likewise, we use the ratio of the probabilities of stepping forward, versus stepping backwards, through the chain, i.e.,

\[
p_{ratio}(x_t, x' ; \mathcal{M}, C) = \frac{p(x' \mid x_t, \mathcal{M}, C)}{p(x_t \mid x', \mathcal{M}, C)}
\]

(8)

We use these two ratios to evaluate

\[
A(x_t, x') = \min \left\{ \frac{q_{ratio}(x_t, x' ; \mathcal{M})}{p_{ratio}(x_t, x' ; \mathcal{M}, C)} \cdot \frac{1}{q_{ratio}(x_t, x' ; \mathcal{M})}, 1 \right\}
\]

(9)

which is the probability with which we accept the candidate state, \( x' \), as the next state, \( x_{t+1} \). Otherwise, we reject \( x' \) and instead set \( x_{t+1} = x_t \).

The first state may be selected arbitrarily, and we discard the first \( \delta \) states to ensure independence, a process known as burn-in. After burn-in, the states are thinned, discarding the \( \delta \) states between each sampled state. This method generates a sequence of pseudo-state \( x_0, \cdots, x_t \) distributed according to probabilities defined in Equation (3). The complete Metropolis-Hastings sampler is shown in Algorithm 1.

Algorithm 1 Metropolis-Hastings for Marginal State Sampling

Input: A model \( \mathcal{M} \), with \( n \) nodes \( \mathcal{V} \) and \( m \) potential edges \( \mathcal{E} \). Current state \( x_t \).

Output: Next state \( x_{t+1} \)

Sample: \( x' \sim q(x_t) \)

Set: \( p_{ratio} = p(x' \mid \mathcal{M}) / p(x_t \mid \mathcal{M}) \)

Set: \( q_{ratio} = q(x' \mid x_t) / q(x_t \mid x') \)

Set: \( \alpha = \min(p_{ratio} / q_{ratio}, 1) \)

Sample: \( r \sim U(0, 1) \)

if \( r \leq \alpha \) then Set: \( x_{t+1} = x' \)

else Set: \( x_{t+1} = x_t \)

Return: \( x_{t+1} \)

We now describe in more detail how to sample from \( q \), and calculate \( q_{ratio} \) and \( p_{ratio} \).
C. The Markov Chain

The Markov Chain in the Metropolis-Hastings sampler selects a new pseudo-state \( x' \) from the current pseudo-state \( x_t \), with probability \( q(x' \mid x_t, M) \). We define the transitions such that the new state \( x' \) differs by at most one edge from \( x_t \), and \( q \) defines a multinomial distribution over edges whose active state will be flipped, from active to inactive or vice versa.

In our experiments, we use transition probabilities proportional to the probability of the resulting activity on the flipped edge. So, in current state \( x_t \), if the point probability of an edge \( e_i \) is \( p_i \), and the edge is inactive, it is selected with probability proportional to \( p_i \), and \( 1 - p_i \) if it is active. If \( x_i \) is the current active status of \( e_i \), then we can define the multinomial distribution over outcomes \( 1 \ldots k \) as

\[
\Delta_t \propto \langle q_1, q_2, \ldots, q_k \rangle,
\]

where outcome \( i \) corresponds to state \( x' \) which differs from \( x_t \) by the activity of edge \( e_i \), and \( q_i = p_i^{x_i}(1 - p_i)^{(1-x_i)} \). We normalize for \( \Delta_t \) with \( Z_t = \sum_i q_i \), and now

\[
q_{\text{ratio}}(x_t, x'; M) = \frac{q(x' \mid x_t, M)}{q(x \mid x', M)} = \frac{p_i^{x_i}(1 - p_i)^{(1-x_i)} Z'}{p_i^{x_i}(1 - p_i)^{(1-x_i)} Z_t}
\]

where \( Z' \) is the normalizing constant of the new state \( x' \). Note that the normalization factor changes by \( 1 - 2p_i \) when we flip an edge \( e_i \). To see this

\[
Z' = Z_t - p_i^{x_i}(1 - p_i)^{(1-x_i)} + p_i^{(1-x_i)}(1 - p_i)^{x_i} = Z_t + (-1)^{x_i}(1 - 2p_i)
\]

We can update the multinomial distribution and take samples in \( O(\log |E|) \) time by constructing a search tree, including updating the normalizing constant.

D. Calculating Pseudo-State Probabilities

We now show how to efficiently evaluate \( p_{\text{ratio}}(x_t, x'; M, C) \). In general, we must ensure the pseudo-state is consistent with flow conditions \( C \in \mathcal{P}(\mathcal{V} \times \mathcal{V} \times \mathcal{B}) \). To this end, the combined indicator function \( I : \mathcal{X} \times \mathcal{P}(\mathcal{V} \times \mathcal{V} \times \mathcal{B}) \to \mathcal{B} \) tests whether a pseudo-state satisfies conditions \( C \)

\[
I(x, C) = \prod_{(u,v,a) \in C} a^l(u,v|x)(1-a)^{1-l(u,v|x)}
\]

Thus, we have

\[
\Pr [x, C \mid M] = I(x, C) \Pr [x \mid M] \tag{7}
\]

This gives the numerator in Equation (6), and thus

\[
p_{\text{ratio}}(x_t, x'; M, C) = \frac{\Pr [x' \mid M, C]}{\Pr [x \mid M, C]} = \frac{\Pr [x', C \mid M]}{\Pr [x_t \mid M, C]} \tag{8}
\]

as the \( \Pr [C \mid M] \) terms cancel out. With the standard \( q \), Equation (8) can be used to sample conditional flow.

Sampling from the Markov Chain, \( q \), takes \( O(\log m) \) time, and testing for the existence of flow in a sampled graph takes \( O(m) \) time. Because we thin the output, discarding samples to ensure independence, each sample takes \( O(\delta'(\log m + m + n \log |C|)) \) time.

E. Uncertainty of Predictions

With a point-probability ICM, there is no uncertainty in the activation probabilities and hence in any derived probability, e.g., \( \Pr [u \sim v] \), if solved exactly. This is no longer true when we have an approximate ICM, such as the betaICM, and we may want to consider the uncertainty in our derived probability. We find this by first sampling an ICM from our betaICM, i.e., \( \forall e_{i,j} \in \mathcal{E} \) sample \( p_{i,j} \sim \text{Beta}(\alpha_{i,j}, \beta_{i,j}) \), then running Metropolis-Hastings on the sampled ICM. By repeat sampling ICMS in a similar MCMC fashion, we can estimate the betaICM’s uncertainty over the flow probability \( \Pr [u \sim v] \), we call this nested Metropolis-Hastings.

IV. EXPERIMENTS ON ATTRIBUTED EVIDENCE

We benchmark our Metropolis-Hastings sampling on synthetic betaICMs, and then test our information flow modeling on data from the Twitter micro-blogging service. We evaluate the effectiveness at predicting information flows, estimating dispersion, refining flow estimates given flow conditions, and evaluating uncertainty in predictions.

A. Synthetic Data Generation

We generate synthetic betaICM by creating a random structure with random edge parameters. Our betaICM generator takes a number of nodes, \( n \); a number of edges, \( m \leq n(n-1) \); and two ranges \([l_a, u_a]\) and \([l_b, u_b]\). The generator creates \( n \) nodes, and adds \( m \) random edges, for each edge \( e \) it draws \( a \sim U(l_a, u_a), b \sim U(l_b, u_b) \) and sets \( B(e) = (a, b) \). For our experiments \( a, b \sim U(1, 20) \).

B. Twitter Micro-Blogging Service

Twitter is a microblogging service that allows users to post short messages, limited to 140 characters, known as tweets. A user follows another user to automatically receive their

\footnote{Using bayesian analysis for conditional probability over unconstrained pseudo-states, we trade off the number of samples with time per sample, \( O(\delta'(\log m + m + n \log |C|)) \).}
tweets in a stream, or the tweets of other users can be viewed from their public timeline (tweet history) or searching. Messages may be forwarded by reposting, known as retweeting. Other users may be referenced in tweets by preceding their name with an ‘@’, e.g., ‘@user123’ refers to user123, and references can be used to reply to users or to indicate the ancestry of a retweet. Authors can also give messages meta-data hashtags in-text by preceding an alphanumeric tag with a ‘#’. In this paper, we are interested in predicting the flow of information via retweets with our methods for attributed data, and predicting the propagation of hashtags and in-message urls through the network, with our methods for unattributed data in Section V.

We use a Twitter dataset gathered by Choudhury et al. ([11]) for our experiments. The data contains 10M tweets and 118K unique users, however, it is sparse and incomplete, containing many retweeted messages without the original tweet. For attributed evidence, we preprocess the tweets, identifying retweets and their attributed parent and possibly more distant ancestors by the message syntax. Searching back through the data, we can link earlier (re)tweets to later retweets, thus building chains of flow of content. We also recover original tweets that are missing. This processes increases the data size to 10.8M tweets and 370K unique users. We omit the details.

C. The Basic Bucket Experiment

To evaluate the overall accuracy of our Metropolis-Hastings sampling approach, we wish to compare the model’s predicted flow probability against more conventionally sampled flow probabilities for a large number of source-sink pairs. Accuracy is tested by what we term our “bucket experiment”, which is adapted from Troncoso and Danzis [12]. Consider that we have a known betaICM, we first sample a point probability ICM from the synthetic betaICM, and an appropriate source and sink node. Next, we obtain an active state from the sampled ICM, and determine if there exists a flow from source-to-sink in this active test state. Using Metropolis-Hastings sampling, we estimate the probability of source-to-sink flow from the betaICM, and pair this with the boolean result from the active test state. We repeat this experiment many times, order the list of pairs by their probability estimates, and partition this list into bins (buckets) of similar predictions. The collection of the boolean parts in each bin helps empirically establish empirical flow probability for the betaICM flow predictions in this bin. We now describe this approach in more detail.

Roughly speaking, we wish to evaluate how frequently an event with estimated probability 0 ≤ x ≤ 1 actually occurs. The entire process is as follows:

1) Generate a synthetic model, \( \mathcal{M} = (V, E, B) \), as discussed.
2) Sample a single active state \( \mathcal{G}_i = (V_i, E_i) \) from \( \mathcal{M} \).
3) Pick \( u, v \in V \) uniform randomly, such that \( u \neq v \).
4) Let \( z = (u \sim v) \) in \( \mathcal{G}_i \), i.e., \( z = 1 \), if and only if there exists a path from \( u \) to \( v \) in \( \mathcal{G}_i \).
5) From \( \mathcal{M} \), estimate \( p_i = \Pr [u \sim v | \mathcal{M}] \). The Metropolis-Hastings sampler can be used to estimate \( p_i \).

6) Bucket pair \( (p_i, z) \in [0, 1] \times \{0, 1\} \) by \( p_i \). The above experiment is repeated a number of times, e.g., thousands. Next, we evaluate how well the estimates \( p_i \) predict the value of \( z \). The pairs \( (p_i, z) \) are divided into \( B \) bins (e.g., 30) of equal size using the estimate \( p_i \). Let \( \text{bin}_j \) be the bin containing \( p_i \in [l_j, u_j) \), such that \( l_j = \frac{j - 1}{B} \) and \( u_j = \frac{j}{B} \). Within each bin, \( \text{bin}_j \), we calculate the mean estimate

\[
\bar{p}_j = \frac{1}{|\text{bin}_j|} \sum_{(p_i, z) \in \text{bin}_j} p_i
\]

The empirical values \( z \), such that \( (p_i, z) \in \text{bin}_j(l_j, u_j) \), are used to construct a beta distribution. For example, when \( z \) is a binary value, testing the flow \( (u \sim v) \), we construct our Beta as follows: \( \alpha_j = 1 + \sum_{(p_i, z) \in \text{bin}_j} z \) and \( \beta_j = |\text{bin}_j| - \alpha_j + 2 \). We construct a beta distribution and confidence interval using the cumulative distribution function for each bucket, and expect the mean estimate \( \bar{p} \) to fall within the 95% confidence interval of the empirical evidence, with approximately 95% chance. Fig. 1 shows results from 2000 synthetic models containing 50 users and 200 edges each. The left figure compares the estimated probability of information flow against the empirical probability. The right plot shows the volume of events and flows. It can be seen that our Metropolis-Hastings estimates are accurate and predominantly within the 95% confidence interval of the empirical data.

The bucket experiment is quite flexible and can be used to evaluate the accuracy of trained betaICMs for predicting real flow probabilities. We begin by training a betaICM from the unattributed retweet evidence extracted from our Twitter data; the network topology is also inferred from the data using the ‘@’ references to indicate edges. We focus on flow between users deemed to be “interesting”, such as those who tweet frequently and whose tweets are retweeted often. For each interesting user we use it as a focus, and a sub-graph of the overall trained model is selected, such that all users are no more than distance \( n \) from this focus. We pick a random sink user, and an empirical sample \( z \) is obtained by testing if the sink retweets a random tweet generated at the source. Then using the trained betaICM and Metropolis-Hastings, we sample a source-to-sink flow probability, \( p \), and use the \( (p, z) \) as a pair for the bucket experiment. Results from 50 random twitter users, no flow conditions, and up to 100 tweets per user, are shown in Fig. 2(a) and 2(b) for sub-graphs with \( n = 1 \) and \( n = 2 \). Further, we experiment by randomly selecting up to five known flows for each real tweet, and use these as flow conditions for the Metropolis-Hastings sampling, see Figures 2(c) and 2(d). Even with noise3, the estimated flow probabilities were within the 95% confidence interval of the empirical samples, performing equally well with conditional flows. For radius 1 retweets, our sampler overestimated some low-end probabilities as seen in Fig. 2(a). A possible explanation for this is that a user may be more likely to retweet an original message than a retweet. Experiments on flow probabilities for path-lengths longer than 3 users are not included as the

3Retweet identification is imperfect and data is incomplete.
Fig. 2: Bucket experiments using attributed evidence from Twitter. For (a)-(d) top (bottom) plots are as respectively described for left (right) plot in Fig. 1.

Fig. 3: Comparing the uncertainty between modeled and empirical flow probabilities. Unbroken line is a beta distribution representing the uncertainty over empirical flow probability in training set. Histogram is of the probabilities sampled from the trained betaICM for the same endpoints. Dashed line is a beta with mean and variance implied by histogram data.

D. Uncertainty and Flow Impact

Next we illustrate how well the betaICM captures the level uncertainty in the evidence. To do this, we select source users, who tweet fairly frequently and in each case randomly choose a nearby sink, see Fig 3 for two examples. We sample roughly 100 ICMs from the betaICM as discussed in Section III-E, and calculate the flow probability for each model. These are compared with empirical Beta distributions trained directly from the same evidence set, i.e. looking at the number of times the source’s tweets are retweeted or not at the sink. These comparisons show that the uncertainty in the original evidence is captured very effectively in the model.

Finally, we use our sampler to estimate the impact of a given tweet as measured by the total number of users who retweet it. Here, we compare the number of retweeting users predicted by the trained betaICM, to the number observed in the separate testing dataset. The results for a random user are shown in Fig. 4. In general, our sampler predicted a similar range of impact, but over estimated the mean impact of a tweet. This may be a product of how the data was originally collected, and we plan to collect data under conditions we can control to test this.

E. Comparison To Random Walk with Restart

An alternative approach to predicting network flows is random walk with restart (RWR) [7], [8]. In [13], RWR was applied to estimate flows in information networks, such as Twitter. There are two problems with RWR: first, it is not sufficiently powerful to estimate the types of joint and conditional flow probabilities or estimate leakage impacts that the ICM model and our Metropolis-Hastings based approach can; and second, RWR is a similarity measure, and not a probability, resulting in less accurate flow estimates. To illustrate, we perform our bucket experiment on synthetic data using RWR, and present the results in Figure 5. When compared to our method in Figure 1, one can clearly see the accuracy improvement.

V. UNATTRIBUTED EVIDENCE

In the previous section we illustrated how to learn a betaICMs from evidence where we have unambiguous knowledge of the attribution of information flowing through the network. There are many instances where attribution information is not available. For example, one may know which users are in possession of information at various points in time, but not the path the information took to get there. We now present a method to learn approximate ICMs from unattributed data, and compare it with a recently published approaches which addresses the same issue [2], [4], [14].

A. Related Estimation Methods

Two recent papers address the training of these models from unattributed evidence. Saito et al. show how to train maximum
Empirical Probability

The parameters have the same meaning in both models. Goyal et al. claim that this approach is not scalable to large data-sets [2], an important factor to us.

There are some important differences between our approach and Saito et al. [4]. First, they assume a time discrete activation process such that if the parent becomes active at time t, the child conditionally activates at only t + 1. In many information networks, such as Twitter, there is no guarantee the child receives information posted at t in step t + 1. We can only be sure that the parent responsible for activating the child was active first. In a followup paper [14], the authors extend ICM to introduce activation delay to address this problem with a significant increase in computation cost. Second, their solution gives point estimates, and cannot calculate the uncertainty or variance in edge probabilities. This is especially important when there is a non-uniform distribution of evidence (e.g., more evidence for one edge than another). Our model uses an informed prior, a Beta distribution, to restrict edge probabilities when accurate prior information is given or inferred from the data. By summarizing the data and using a Binomial distribution instead of a set of Bernoulli variables, we can significantly reduce the computational costs at each step, such as calculating the model fitness or performing a maximization step (if we were to choose an EM approximation).

In addition to our joint Bayes solution, we modify the Saito EM approximation to relax the assumption on activation times, consistent with assumptions made in [2] and this paper. We illustrate how to perform the E and M steps using summarized evidence to reduce computation costs in the Appendix.

EM solutions can be susceptible to local-maxima, and solutions such as random restart or annealing make the approach more expensive. Also, as we are maximising the likelihood function, we are only finding the mode (not the mean) of our distribution. An example is provided in the Appendix. This can be a poor substitute with multi-model distribution, and as we show the conditions for such a distribution exist in our models.

Goyal et al. [2] present a more scalable approach for approximating activation (diffusion) probabilities from unattributed evidence. They describe this in the context of General Threshold Models (GTMs) of information flow, but then demonstrate their method on a subclass of GTMs, we call Simplified (SGTMs), where each parent’s influence is fixed. We now describe SGTMs and show they are equivalent to ICMs, and that the parameters have the same meaning in both models.

In SGTMs, the mechanism by which a node becomes active is described differently. For each object, o, and each node, v, a random threshold probability, \( p_{o,v} \in [0, 1] \), is drawn uniformly randomly. If the active parents of v at time t, then the influence probability from \( S_t \) on v is given by \( p_v(S_t) \), where

\[
p_v(S_t) = 1 - \prod_{u \in S_t} (1 - p_{u,v})
\]

where \( p_{u,v} \) is the weight on edge \((u, v)\) and \( S_t \subseteq S_{t+1} \). v becomes active at the earliest time t at which \( p_v(S_t) > p_{o,v} \).

**Theorem 1:** The two models SGTM and ICM are equivalent, and the weights \( p_{u,v} \) mean the same thing in both models.

**Proof:** If \( v \) is active given the set \( S \) of all active parents, we write \((S \rightarrow v)\); otherwise, we write \((S \not\rightarrow v)\). For SGTMs, the probability that \( v \) is not active when parents \( S \) are active, then becomes active when the next parent, \( w \), becomes active, is given, for \( S' = S \cup \{w\} \), as

\[
\Pr[(S' \rightarrow v) \land (S \not\rightarrow v) \mid \text{SGTM}] = \Pr[p_v(S) < p_{o,v} \leq p_v(S')] = p_v(S') - p_v(S) = \prod_{u \in S} (1 - p_{u,v}) - \prod_{u \in S'} (1 - p_{u,v}) = p_{w,v} \prod_{u \in S} (1 - p_{u,v})
\]

For ICMs the probability of the same event is given by

\[
\Pr[(S' \rightarrow v) \land (S \not\rightarrow v) \mid \text{ICM}] = \Pr[(S' \rightarrow v) \mid \text{ICM}] \Pr[(S \not\rightarrow v) \mid \text{ICM}] = p_{w,v} \prod_{u \in S} (1 - p_{u,v})
\]

Thius for edge \((u, v)\), Goyal et al. are approximating the same parameter, \( p_{u,v} \). We next describe the two approaches.

**B. Training from Unattributed Evidence**

Without loss of generality, assume we wish to calculate the leakage probabilities for all edges incident on some sink node, \( k \). In the attributed case, we assume \( p_{j,k} \sim \text{Beta}(\alpha_{j,k}, \beta_{j,k}) \), where \( \alpha_{j,k} \) and \( \beta_{j,k} \) are easily inferred from the data by counting the number of positive and negative events, plus one, respectively. In the unattributed case, for an object \( o \), there may be multiple nodes, \( J_o = \{j_0, j_1, \ldots, j_l\} \) that are active temporally before \( k \), and each may leak the information to \( k \). When \( |J_o| = 1 \), we have unambiguous attribution.

When \( |J_o| > 1 \), each incident node may have leaked the information to \( k \). Goyal et al. [2] present a method where each node \( j \in J_o \) is assumed to have equally contributed to \( k \)'s activation, i.e.,

\[
\text{credit}_{k, J_o}(o) = \frac{k_o}{|J_o|}
\]

and an edge’s trained activation probability is the sum of its credit normalized by the number of times the parent was active,

\[
p_{j,k} = \frac{\sum_{o \in \mathcal{O}} \text{credit}_{j, J_o}(o)}{|\{o \mid j \in J_o\}|}
\]

This approach is only a rule of thumb, and can result in a biasing activation probabilities towards the mean of all edges
TABLE I: Example Evidence Summary for sink \(k\) with incident nodes \(A\), \(B\), and \(C\).

incident to \(k\). We next show a more principled generative approach to training activation probabilities to obtain more accurate estimates and capture the uncertainty in the edge probabilities as we did for the attributed case.

We call the set of users active before \(k\) for a given object \(o\), \(J_o\), the characteristic, easily expressed as a bit vector. Given evidence as activation times for a set of objects, we can define a summary for our evidence \(D\) as for each \(k\): the set of unique characteristics, the number of times each characteristic is observed, and the number of times each characteristic resulted in \(k\) becoming active. Note that, if \(k\) becomes active for \(o\), then the observed characteristic is the active characteristic just prior to \(k\) being active; otherwise it is the active characteristic at the latest time in the data. This ensures that all potential causes are considered for both positive and negative flows. A summary is a sufficient statistic for our model (as we shall show), and hence takes the place of evidence. An example summary for a four-node graph is given in Table I.

We seek the model, \(M = (V, E, \mathcal{P})\), with activation probabilities, \(\mathcal{P}\), that best explains the evidence observed. For clarity, we partition the model by edges, where each part is a model \(M_k\) with only those edges incident on node \(k \in V\) and all incident nodes; from the ICM’s assumption the evidence that relates to this model part is the summary for \(k\), \(D_k\). This is a generative model, and we begin by defining the likelihood of the model given the evidence in terms of the likelihood of the evidence given the model, so

\[
p(M | D) = \prod_k p(M_k | D_k) = \prod_k \frac{\Pr[D_k | M_k] p(M_k)}{\Pr[D_k]}
\]

and we can train each model part independently.

The probability \(\Pr[D_k | M_k]\) can be efficiently calculated directly from the independent cascade model because all flows are atomic. Given sink \(k\), model \(M_k\) with activation probabilities \(p_{j,k}\), and a characteristic \(J\), the probability that \(k\) is activated is \(p_{J,k} = 1 - \prod_{j \in J} (1 - p_{j,k})\). To observe \(n_J\) events of characteristic \(J\), of which \(L_J\) have resulted in information flowing to \(k\) in \(D_k\) we must sample from the binomial distribution \(L_J \sim \text{Binomial}(n_J, p_{J,k})\), i.e.,

\[
\Pr[L_J | n_J, M_k] = \binom{n_J}{L_J} p_{J,k}^{L_J} (1 - p_{J,k})^{n_J - L_J}
\]

and for all characteristics in the evidence

\[
\Pr[D_k | M_k] \propto \prod_{J \in D_k} \Pr[L_J | n_J, M_k]
\]

\[
\propto \prod_{J \in D_k} \binom{n_J}{L_J} p_{J,k}^{L_J} (1 - p_{J,k})^{n_J - L_J}
\]

For the marginal probability density, \(p(M_k)\), we assume the probabilities in model \(M_k\) are drawn from Beta distributions. Let \(p_{j,k} \sim \text{Beta}(\alpha_{j,k}, \beta_{j,k})\), where \(\alpha_{j,k}\) and \(\beta_{j,k}\) are calculated from the unambiguous characteristics only, i.e., there was a single active incident node. Note that when there is no evidence from the characteristic define by \(j\) then we assume the default prior \(\text{Beta}(1,1)\), i.e., the uniform distribution. Therefore,

\[
p(M_k) = \prod_j \text{Beta}(p_{j,k}; \alpha_{j,k}, \beta_{j,k})
\]

where

\[
\text{Beta}(x; \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 u^{\alpha-1}(1-u)^{\beta-1} du}
\]

Because we do not know the normalization factor, we estimate the posterior distribution of the edge probabilities, \(p_{j,k}\), using the activation times as evidence, using the Metropolis-Hastings algorithm. Estimating edge probabilities was implemented using the PyMC package [15] in about 50 lines of Python (not including aggregating evidence).

C. Analysis

We now evaluate the performance and accuracy of our method compared to Goyal et al.’s at learning correct edge probabilities. The complexity of both methods is \(O(nm)\) where \(n\) is the number of incident nodes and \(m\) is the number of objects when we don’t summarize the data. If we do perform summarization, the complexity is \(O(nw)\), where \(w\) is the number of unique characteristics and \(\omega = O(\min(2^n, m))\); in practice it is much less. For our approach this entails a tradeoff between \(m\) Bernoulli trials and \(\omega\) Binomials plus the cost of summarization. The main computation difference between the running time of the two approaches are hidden by the constants; Goyal’s approach requires \(m + n\) divisions and \(mn\) additions, while our requires calculating the log likelihood of \(n\) Betas, and \(\omega\) Binomials. The difference in the running time to draw a single sample of the core computation of the two approaches is given in Figure 6(a), and the total time in Figure 6(b) (one sample plus summarization in dots, and the amortized cost per sample in crosses).

For the marginal probability density, \(p(M_k)\), we assume the probabilities in model \(M_k\) are drawn from Beta distributions. Let \(p_{j,k} \sim \text{Beta}(\alpha_{j,k}, \beta_{j,k})\), where \(\alpha_{j,k}\) and \(\beta_{j,k}\) are calculated from the unambiguous characteristics only, i.e., there was a single active incident node. Note that when there is no evidence from the characteristic define by \(j\) then we assume the default prior \(\text{Beta}(1,1)\), i.e., the uniform distribution. Therefore,

\[
p(M_k) = \prod_j \text{Beta}(p_{j,k}; \alpha_{j,k}, \beta_{j,k})
\]

where

\[
\text{Beta}(x; \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 u^{\alpha-1}(1-u)^{\beta-1} du}
\]

Because we do not know the normalization factor, we estimate the posterior distribution of the edge probabilities, \(p_{j,k}\), using the activation times as evidence, using the Metropolis-Hastings algorithm. Estimating edge probabilities was implemented using the PyMC package [15] in about 50 lines of Python (not including aggregating evidence).

C. Analysis

We now evaluate the performance and accuracy of our method compared to Goyal et al.’s at learning correct edge probabilities. The complexity of both methods is \(O(nm)\) where \(n\) is the number of incident nodes and \(m\) is the number of objects when we don’t summarize the data. If we do perform summarization, the complexity is \(O(nw)\), where \(w\) is the number of unique characteristics and \(\omega = O(\min(2^n, m))\); in practice it is much less. For our approach this entails a tradeoff between \(m\) Bernoulli trials and \(\omega\) Binomials plus the cost of summarization. The main computation difference between the running time of the two approaches are hidden by the constants; Goyal’s approach requires \(m + n\) divisions and \(mn\) additions, while our requires calculating the log likelihood of \(n\) Betas, and \(\omega\) Binomials. The difference in the running time to draw a single sample of the core computation of the two approaches is given in Figure 6(a), and the total time in Figure 6(b) (one sample plus summarization in dots, and the amortized cost per sample in crosses).

![Fig. 6: Time for our method compared to Goyal’s to draw a single sample in seconds.](image)

To evaluate the accuracy and data efficiency of our method and Goyal at al.’s, we train models on synthetic data generated from randomly generated ICMs (the ground truth).
The activation probabilities on the ground truth graphs are skewed, i.e. of two types: 90% are drawn from Beta(16, 4) – mean 0.8 and narrow distribution; 10% are drawn from Beta(2, 8) – mean 0.2 and wider distribution. We focus on each method’s accuracy in learning activation probabilities for edges incident on a single node; this is indicative of the overall accuracy performance. In Figure 7, we plot the root mean squared error between the trained models, and the ground truth. We also include the RMSE betaICMs trained filtered with the attributed method using only those objects where attribution is unambiguous (i.e. a single active parent), and with the attributed method using only those objects where ground truth. We also include the RMSE betaICMs trained for edges incident on a single node; this is indicative of each method’s accuracy in learning activation probabilities (some edges are positively or negatively correlated).

The extra computational costs of our approach, therefore yields two significant advantages. First, our solution is more accurate, resulting in a decreased error on the estimated edge probabilities which we can measure when ground truth is known. The advantages are especially pronounced when where is a large skew between activation probabilities across multiple edges. Second, our approach measures the amount of uncertainty in the edge probabilities (and can even indicate if some edges are positively or negatively correlated).

![Graph Showing RMSE](image)

**Fig. 7: Root mean squared error of trained graph fragments versus ground truth.** Models trained on synthetic data with ground truth graphs generated randomly. Activation probabilities shown for each plot: without skew (a) and (c); with skew (b) and (d). Dashed lines show 95% confidence interval of uncertainty on joint bayes.

### D. Predicting Hashtags and URLs in Twitter

In this section we evaluate the performance of our method for learning unattributed flows and estimating flow probabilities to predict the propagation of objects of finer granularities than tweets in Twitter. For each tweet, we identify the hashtags in the tweet (hashtags are prefixed by a pound, #, and denote topics), and URLs, typically shortened. We learn edge probabilities on these types independently, i.e., for each edge we learn a retweet probability, a hashtag probability, and a URL probability. We select a set of “interesting” users that are the originators of many popular hashtags and URLs.

Because hashtags and URLs can come from outside of Twitter, e.g., real-world events, blogs, news and radio programs, etc., we define an omnipotent user to express the outside world. All users follow this hypothetical entity, and is the true originator of all tweets. We then learn edge probabilities for propagating hashtags and URLs for social graphs flowing outward from a user of interest. Because our approach calculates a full joint probability distribution over all edges, we store an approximation using the mean and standard deviation. For simplicity, we use an edge-independent mean for all experiments. We then calculate the flow probability from the source user to every user in the limited social graph.

Our analysis indicates that hashtags and URLs have different flow characteristics in Twitter, resulting in poor estimates of flows of hashtags using edge probabilities learned using either our method or Goyal et al.’s for edges, and using the independent cascade model. First, Figure 8 illustrates the performance of predicting flows of URLs for social networks of radius 4 and 5 hops from the source user (not including the omnipotent user). Here, we can see that in practice our model for learning edge probabilities is more accurate, validating the observation made on synthetic graphs (Figure 7). When we omit the omnipotent user that represent flow outside of Twitter, we find the flow probabilities are increased marginally.

More interesting is the substantially poorer performance at predicting flows of hashtags, using either method, show in Figure 9. We believe this is indicative of how hashtags enter Twitter compared to URLs, and the entropy of each. Hashtags are topical tags that often mark events coordinated offline (e.g., a prior agreed common tag for an event #ICDE), a low entropy acronym many users will independently discover (e.g., #POTUS), a convention to trigger an external event (e.g., #FB for FaceBook status updates), etc. This contrasts significantly with URLs that are often randomly generated with a URL shortener such as bit.ly. Here, a shortened URL may have many forms and users are unlikely to tweet them outside without receiving it previously in their Twitter timeline. Finally, because our method can capture the amount of uncertainty in the edge probabilities, we sample 30 graph independently, i.e., for each flow we obtain a distribution of flow probabilities, and not a point estimate. We sample each edge independently using its mean and standard deviation from a normal distribution. A more accurate alternative would be to store samples from the full joint distribution that captures dependencies between multiple edges. We find when taking the edge-uncertainty into consideration has a smoothing effect on flow probabilities, as shown in Figure 10. Note that this has the side effect of placing fewer points into each bucket, leading to an increased uncertainty in the empirical estimates.
VI. DISCUSSION

As exact flow calculations in the popular Independent Cascade information flow model (ICMs) are exponential, we present an approximation using Metropolis-Hastings sampling. We illustrate the accuracy of our model on retweet data from Twitter. ICMs can calculate joint and conditional flow probabilities, and unlike other models, do not require any simplifying assumptions such as acyclic flow to remain tractable. Further, our model allows us to calculate not just a point probability, but a distribution over flow probabilities. Experiments show that uncertainty in evidence is mirrored in the trained models. This might be considered surprising as each activation probability estimate between source and sink is trained using tweets from multiple sources, and supports the assumption underlying ICMs, that activation probabilities based only on the edge can capture a significant portion of the flow behaviour. This has many interesting applications, such as risk-aware calculations of information leakage.

We also present a new approach for learning edge probabilities without full attribution where one does not know the parent node that caused a child to activate. This allows us to expand information flow to finer granularities, such as hashtags and
URLs. We illustrated our approach was practical, and accuracy is improved compared to competing methods or does not have the same shortcomings.

We plan on extending our model to include edge activation probabilities that depend on context, e.g., using different retweet distributions when not quoting the originating user. Other extensions include adding edge latency or delay before a message is forwarded. This is trivially solved by assigning a delay distribution to each edge, and sample from these distributions for each sample from the posterior, i.e., assigning a weight to each edge that represents a time, and running a shortest path algorithm. This is in contrast to the extension to a weight to each edge that represents a time, and running a shortest path algorithm. This in in contrast to the extension to ICM from Saito et al. [14]. Other extensions include modeling the amount of information that is forwarded, at granularities other than tweet, hashtag, or URL. This requires extracting finer granularity features from objects, and an information divergence between two tweets, such as Kullback-Leibler. For example, a user may rewrite the tweet, keeping the content intact, i.e., plagiarism. We leave these for future work.

REFERENCES


APPENDIX

We noted in Section V-A, Saito et al. [4] is an expectation maximization approach and may converge on a local maximum. A local maximum may be a problem when the distribution is multimodal, requiring other solutions, such as random restart or annealing. Further, an EM solution only provides a single point, the mode, for each restart. This does not provide information on the potential spread or uncertainty in the point estimates given the evidence. To illustrate, we randomly restart Saito et al.’s algorithm 1000 times on a small example shown in Table II, and we run our joint Bayes solution using MCMC once, and plot 1000 samples.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Count</th>
<th>Leaks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>100</td>
<td>75</td>
</tr>
</tbody>
</table>

TABLE II: Example graph with three incident nodes, A, B, and C, and resulting in a multimodal distribution of edge probabilities.

Fig. 11: The EM solution only finds local maxima. Fixing Saito at 200 iterations.

Saito et al.’s EM approach assumes that for a parent to cause a child to be active, then the implicated parent must be active in the preceding time-step. Our EM implementation relaxes this and only assumes that implicated parents are active before the relevant child. We now demonstrate how this assumption with our summarisation approach change the expectation and maximisation steps from [4]. The expectation step, equivalent to [4, Equation (6)], is

\[
\hat{\mathbf{P}}_{w}^{\mathcal{J}} = 1 - \prod_{v \in \mathcal{J}} (1 - \hat{\kappa}_{v,w})
\]

The maximisation step, equivalent to [4, Equation (8)], is

\[
\hat{\kappa}_{v,w} = \left\{ \begin{array}{ll}
\frac{\hat{\kappa}_{v,w}}{|S_{v,w}^{+}| + |S_{v,w}^{-}|} \sum_{i \in \mathcal{J}} \frac{L_{i}}{P_{i}^{w}} & \text{if } S_{v,w}^{+} > 0 \\
\hat{\kappa}_{v,w} & \text{otherwise}
\end{array} \right.
\]

where \( S_{v,w}^{+} = \{ i \in \mathcal{O} | v, w \in \mathcal{V}_{i} \} \) and \( S_{v,w}^{-} = \{ i \in \mathcal{O} | v \in \mathcal{V}_{i}, w \notin \mathcal{V}_{i} \} \). We omit details of the derivation.
Accuracy Measures

I looked into a number of potential measures of success, but I have limited things to the two most significant. The first is the geometric mean of the probability of an outcome given the prediction, which a normalised measure of the likelihood of the data given the predictions and so I call it the normalised likelihood. For normalised likelihood the closer to 1 the better. The second is called the Brier probability score, and is essentially the mean square difference between the prediction (a probability) and the outcome (a boolean). For the Brier probability we are interested in values being closest to 0. The normalised likelihood gave erroneous results when the prediction was exactly 1 or 0, as the likelihood of getting the less likely value (and hence the entire dataset) becomes 0, so I modified these values to be not quite 1 or 0. The table below shows the results.

The was some difficulty pulling apart the methods MC and Goyal with the full dataset, this seemed to be that the vast majority of predictions were very close to 0 (and in fact were in the database as exactly 0 which I assumed was a rounding error). To avoid these values washing out the differences on the other datapoints, I reran the experiments but ignored all predictions which were exactly 0 or 1, these also appear in the table and are labelled middle values.

<table>
<thead>
<tr>
<th>exp.</th>
<th>Normalised Likelihood</th>
<th>Brier Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>all values (count)</td>
<td>middle values (count)</td>
</tr>
<tr>
<td>MH Test – Fig. 1</td>
<td>0.598721</td>
<td>0.575576</td>
</tr>
<tr>
<td>RWR – Fig. 5</td>
<td>0.350548</td>
<td>0.350108</td>
</tr>
<tr>
<td>Which of these are for Fig. 2?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>exp_r1_g500_z100.db</td>
<td>0.95061</td>
<td>0.937024</td>
</tr>
<tr>
<td>exp_r1_g50_z100.db</td>
<td>0.958829</td>
<td>0.925418</td>
</tr>
<tr>
<td>exp_r2_g20_z20.db</td>
<td>0.990100</td>
<td>0.980100</td>
</tr>
<tr>
<td>exp_p2_r50_z100.db</td>
<td>0.985531</td>
<td>0.971837</td>
</tr>
<tr>
<td>net3_r1_g50_z100.db</td>
<td>0.970823</td>
<td>0.968018</td>
</tr>
<tr>
<td>net3_r2_g50_z100.db</td>
<td>0.984728</td>
<td>0.986144</td>
</tr>
<tr>
<td>MC (radius 4) – Fig. 8(a)</td>
<td>0.985377</td>
<td>0.902585</td>
</tr>
<tr>
<td>Goyal (radius 4) – Fig. 8(c)</td>
<td>0.984493</td>
<td>0.601576</td>
</tr>
<tr>
<td>MC (radius 5) – Fig. 8(b)</td>
<td>0.996066</td>
<td>0.910000</td>
</tr>
<tr>
<td>Goyal (radius 5) – Fig. 8(d)</td>
<td>0.996544</td>
<td>0.822394</td>
</tr>
</tbody>
</table>

TABLE III: Some performance measures

It is worth noting that the performance can be much higher when predictions are near certain. Therefore when these near certain results are removed, the performance measure almost certainly will decrease. Moreover, the change is most profound, if there are lots of predictions at near certainty. Therefore, it should not be necessarily seen as a bad thing if the performance measure is different between all values and middle values.