CSI: Community-Level Social Influence Analysis

Yasir Mehmood[1], Nicola Barbieri[2], Francesco Bonchi[2], and Antti Ukkonen[3]

[1] Pompeu Fabra University, Spain
yasir.mehmood01@estudiant.upf.edu
[2] Yahoo! Research Barcelona, Spain
{barbieri,bonchi}@yahoo-inc.com
[3] Helsinki Institute for Information Technology HIIT, Aalto University, Finland
antti.ukkonen@aalto.fi

Abstract. Modeling how information propagates in social networks driven by peer influence, is a fundamental research question towards understanding the structure and dynamics of these complex networks, as well as developing viral marketing applications. Existing literature studies influence at the level of individuals, mostly ignoring the existence of a community structure in which multiple nodes may exhibit a common influence pattern.

In this paper we introduce CSI, a model for analyzing information propagation and social influence at the granularity of communities. CSI builds over a novel propagation model that generalizes the classic Independent Cascade model to deal with groups of nodes (instead of single nodes) influence. Given a social network and a database of past information propagation, we propose a hierarchical approach to detect a set of communities and their reciprocal influence strength. CSI provides a higher level and more intuitive description of the influence dynamics, thus representing a powerful tool to summarize and investigate patterns of influence in large social networks. The evaluation on various datasets suggests the effectiveness of the proposed approach in modeling information propagation at the level of communities. It further enables to detect interesting patterns of influence, such as the communities that play a key role in the overall diffusion process, or that are likely to start information cascades.

1 Introduction

Understanding the dynamics of influence in online social networks is becoming an interesting point of convergence for different subjects, including social science, statistical analysis and computational marketing. Social influence analysis is receiving a growing attention by both academic and industrial communities, mainly due to the wide range of applications, e.g personalized recommendations, viral marketing, feed ranking, and scenarios in which influence plays an important role in predicting users' behavior. Most of the networks of interests for this analysis are very large, with millions of edges. Therefore, *graph summarization* techniques are needed in order to help the analysis by highlighting the main properties of the influence dynamics and recurring patterns. Most of the research in graph summarization has focused on finding abstraction of a graph (e.g., by aggregating nodes in meta-nodes) that preserves the structural properties of the original graph, or properties defined as aggregates over the node attributes. In this

H. Blockeel et al. (Eds.): ECML PKDD 2013, Part II, LNAI 8189, pp. 48–63, 2013.

paper instead, our goal is to devise a graph summarization paradigm for the analysis of the phenomena of information propagation and social influence. More in concrete, we aim to find an abstraction which, although being coarser than the original data, it still describes well a database of past propagation traces. Our technique provides the data analyst with a compact, and yet meaningful, view of the patterns of influence and information diffusion over the considered network, where members of the same community tend to play the same role in the information propagation process.

Towards this goal, we extend the well known Independent Cascade model [10], to study influence at the level of communities. Briefly, the community structure detected by our approach reflects macro influence propagation patterns. A community is identified by a set of *connected nodes* that share a *similar influence tendency* over nodes belonging to other communities. The strength of influence relationships between communities can be used to understand the importance of their connection. Moreover, by directly modeling community-level influence relationships, we can provide an high level picture of the global diffusion process over the network, and a summary of the main influence patterns that shape the underlying process of information propagation.

The main contributes of this paper are the following:

- We introduce the CSI (**C**ommunity-level **S**ocial **I**nfluence) model, which extends the peer-influence relationships that define the Independent Cascade model at the granularity of communities.
- We devise a greedy algorithm which explores a given hierarchical partitioning of the network and provides as output the community structure that achieves a good balance between the accuracy in describing observed propagation data, and a compact representation of the influence relationships.
- Given a set of disjoint communities, we devise an Expectation-Maximization algorithm to effectively learn the strength of their pairwise influence relationships.
- Through an experimental evaluation on three real-world datasets, we show the effectiveness of our approach, which is able to provide a meaningful and compact summary of the influence patterns on the considered networks.

The rest of the paper is organized as follows. We briefly review related prior art in Sec. 2. In Sec. 3, we formally define the problem tackled in this paper while our algorithm is presented in Sec. 4. The experimental evaluation is in Sec. 5. Finally, Sec. 6 concludes the paper with a summary of our major findings and future direction of research.

2 Related Work

Social influence and the phenomenon of influence-driven propagations in social networks have received considerable attention in the last years. One of the key problems in this area is the identification of a set of influential users in a given social network. Domingos and Richardson [4] approach the problem with Markov random fields, while Kempe et al. [10] frame influence maximization as a discrete optimization problem. Another vein of study has focused on the problem of learning the influence probabilities on every edge of a social network given an observed log of propagations over this network [16,18,7,20]. In this paper we use the method by Saito et al. [16].

Although our goal is that of summarizing the social graph, our work could also be collocated in the wide *community detection* literature: for a thorough survey of the topic we refer the reader to [5]. While the bulk of this literature only focuses on the structure of the social graph, a recent paper [1] is the first to define a community-detection mechanism that exploits information propagation traces to find better communities. Our contribution in this paper is different as we aim at modeling community-to-community influence, while the goal of [1] is to find good communities w.r.t. the graph structure and information propagation.

Finally, many tasks in machine learning and data mining involve finding simple and interpretable models that nonetheless provide a good fit to observed data. In graph summarization the objective is to provide a coarse representation of a graph for further analysis. Tian et al. [19] as well as Zhang et al. [21] consider algorithms to build graph summaries based on node attributes, while Navlakha et al. [13] use MDL to find good structural summaries of graphs. In [14] this method is applied to study protein interaction networks. Our work is also related to research that uses a taxonomy to impose the right level of granularity to the model being learned. Garriga et al. [6] consider the problem of feature selection for regression models given a taxonomy over the independent variables, while Bonchi et al. [2] use a hierarchical decomposition a state space to simplify Markov models. Lavrač et al. [11] construct interpretable rules by selecting attributes with the help of a hierarchical ontology.

3 Community-Level Social Influence Model

We first (Sec. 3.1) recall the independent cascade propagation model [10], that is at the basis of our proposal. Then we introduce CSI (Sec. 3.2), our model that generalizes peer-influence to the community level: we devise the procedure for learning the parameters of the model (Sec. 3.3), and we discuss model selection (Sec. 3.4).

3.1 Preliminaries: The Independent Cascade (IC) Model

Let $G = (V, E)$ denote a directed network, where V is the set of vertices and $E \subseteq V \times V$ denotes a set of directed arcs. Each arc $(u, v) \in E$ represents an influence relationship, i.e u is a potential influencer for v, and it is associated with a probability $p(u, v)$ representing the strength of such influence relationship. Let $\mathbb{D} = \{\alpha_1, \cdots, \alpha_r\}$ denote a log of observed propagation traces over the G. For each trace α the log contains the activation time of each node $t_\alpha(v)$, where $t_\alpha(v) = \infty$ if v does not become active in trace α.

We assume that each propagation trace in $\mathbb{D}$ is initiated by a special node $\Omega \notin V$, which models a source of influence that is external to the network. More specifically, we have $t_\alpha(\Omega) < t_\alpha(v)$ for each $\alpha \in \mathbb{D}$ and $v \in V$. Time unfolds in discrete steps. At time $t = 0$ all vertices in V are inactive, and Ω makes an attempt to activate every vertex $v \in V$ and succeeds with probability $p(\Omega, v)$. At subsequent time steps, when a node u becomes active, it makes one attempt at influencing each inactive neighbor v with probability $p(u, v)$. Multiple nodes may try to activate, independently, the same node at the same time. If at least one attempt to activate node v at time t succeeds, then

v becomes active at $t+1$. Therefore, at a given time step, each node is either active or inactive, and active nodes never become inactive again.

Note that we have not specified the function p in detail. The independent cascade model can be instantiated with an arbitrary choice of p, but these come with different trade-offs. Kempe et al. [10] use a uniform probability q in their experiments, that is, $p(u,v) = q$ for all $(u,v) \in E$. On the other hand, Saito et al. [16] estimate a separate probability $p(u,v)$ for every $(u,v) \in E$ from a set of observed traces. These two approaches can be viewed as opposite ends of a complexity scale. Using a single parameter leads to a simple but potentially inaccurate model, while estimating a different probability for each arc might provide a good fit but at the price of risking to overfit, due to the very large number of parameters [12,8].

Next we introduce our CSI model, that shifts the modeling of influence strength from node-to-node, to community-to-community. In our community-based variant of the IC model, all vertices that belong to the same cluster are assumed to have identical influence probabilities towards other clusters.

3.2 The CSI Model

We start by introducing the likelihood of a single trace α when expressed as a function of single edge probability: this is needed to define the problem that we tackle in this paper.

When it comes to fit real data, some of the assumptions of the the theoretical IC propagation model are hardly met. For instance time is not coarsely discrete, and we cannot assume that in real data, if u activates at time t and it succeeds in influencing some of its peers, then this will happen at time $t+1$. Following [12], we circumvent this problem by adopting a *delay threshold* Δ to distinguish between potential influencers that may have triggered an activation, and those who have certainly failed. Let $F^{+}_{\alpha,u}$ be the set of u's neighbors that potentially influenced u's activation in the trace α:

$$F^{+}_{\alpha,u} = \{v \mid (v,u) \in E, 0 \leq t_\alpha(u) - t_\alpha(v) \leq \Delta\}.$$

Similarly, we define as $F^{-}_{\alpha,u}$ the set of u's neighbors who definitely failed in influencing u on α:

$$F^{-}_{\alpha,u} = \{v \mid (v,u) \in E, t_\alpha(u) - t_\alpha(v) > \Delta\}.$$

Let $p : V \times V \to [0,1]$ denote a function that maps every pair of nodes to a probability. The log likelihood of the traces in $\mathbb{D}$ given p can be expressed as

$$\log L(\mathbb{D} \mid p) = \sum_{\alpha \in \mathbb{D}} \log L_\alpha(p), \tag{1}$$

because the traces are assumed to be i.i.d. The likelihood of a single trace α is

$$L_\alpha(p) = \prod_{v \in V} \left[1 - \prod_{u \in F^{+}_{\alpha,v}} (1 - p(u,v))\right] \cdot \left[\prod_{u \in F^{-}_{\alpha,v}} (1 - p(u,v))\right]. \tag{2}$$

As we already anticipated, in the CSI model we shift the influence strength estimation from node-to-node, to community-to-community. To this end, we use a hierarchical

decomposition $\mathcal{H}$ of the underlying network G. In particular, $\mathcal{H}$ is a *tree* rooted at r, with the nodes in V as leaves, and an arbitrary number of internal nodes. A *cut* h of $\mathcal{H}$ is a set of edges of $\mathcal{H}$, so that for every $v \in V$, one and only one edge $e \in h$ belongs to the path from the root r to v. Therefore, the removal of the edges in h from $\mathcal{H}$ disconnects every $v \in V$ from r.

Let $C(\mathcal{H})$ denote the set of all possible cuts of $\mathcal{H}$. Each $h \in C(\mathcal{H})$ induces thus a partition P_h of the network G, so that all vertices in V that are below the same edge $e \in h$ in $\mathcal{H}$ belong to the same cluster $c_e \subseteq V$. Let $a(v)$ denote the cluster to which the node $v \in V$ belongs to in the partition P_h.

In the CSI model, all vertices that belong to the same cluster are assumed to have identical influence probabilities towards other clusters. Given a function $\tilde{p}_h : P_h \times P_h \rightarrow [0, 1]$ that assigns a probability between any two clusters of the partition P_h, we define

$$p_h(u, v) = \tilde{p}_h\left(a(u), a(v)\right).$$

Below, in Section 3.3, we will show that given G, $\mathcal{H}$, the cut h, and $\mathbb{D}$, we can find $\tilde{p}_h$ using an expectation maximization (EM) algorithm. For the moment we can assume that $\tilde{p}_h$ is induced by h in a deterministic way, because our aim is to define our problem in terms of finding an optimal cut $h^* \in C(\mathcal{H})$.

A straightforward observation is that the likelihood defined in equations 1 and 2 is maximized by the cut at the leaf level of $\mathcal{H}$. Reducing the number of pairwise influence probabilities used by the model can only result in a lower likelihood. Therefore, we use a *model selection function* f that takes into account both likelihood as well as the complexity of the model. We discuss choices for f later in Section 3.4. Note that since the network G and the hierarchy $\mathcal{H}$ remain fixed, model complexity is only affected by the cut $h \in C(\mathcal{H})$.

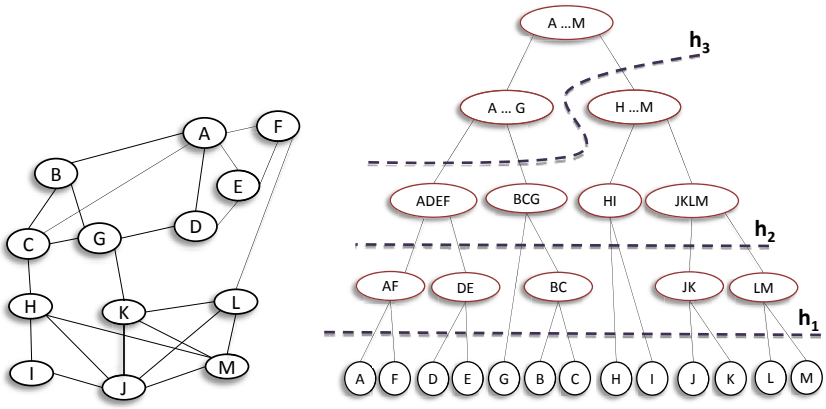
Example 1. Figures 1 and 2 illustrate a possible input for our problem and a possible output, i.e., a CSI model, respectively. In particular in the example, the cut h_1 corresponds to the leaf level model where each single node of the social graph constitutes a state of the CSI model: this is the maximum likelihood cut. However, this would correspond to the standard IC model and is not our goal. In the picture two other cuts are shown, where h_2 corresponds to the clustering $\{\{A, F\}, \{D, E\}, \{BC\}, \{H\}, \{I\}, \{J, K\}, \{L, M\}\}$, and the cut h_3 results in the CSI model in Figure 2 , which in our example is the *"best"* model according to the model selection function f.

Now we have all necessary ingredients to formally state the problem addressed in this paper.

Problem 1 (Learning a CSI model). Given a network $G = (V, E)$, a log of propagation traces $\mathbb{D}$ across this network, a hierarchical partitioning $\mathcal{H}$ of G, and a model selection function f, find the optimal cut of $\mathcal{H}$ defined as

$$h^* = \arg\min_{h \in C(\mathcal{H})} f\left(L(\mathbb{D}, p_h), h\right).$$

We do not formally address the complexity of the problem in this paper. An exhaustive enumeration of all possible cuts is infeasible, since the size of $C(\mathcal{H})$ can be



$\mathbb{D}$

Ω	A	B	M	Ω	B	C	G	K	. . . . .	Ω	A	L	M	C
0	40	45	50	0	10	15	23	47	. . . . .	0	20	48	53	97

Fig. 1. An example of input for our problem: a social graph G (here represented as undirected, but we can always consider each undirected edge as the corresponding two directed arcs), a hierarchy $\mathcal{H}$ over G, and a log $\mathbb{D}$ of observed propagation traces over the social graph G

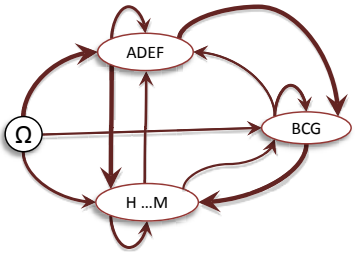


Fig. 2. A possible CSI model resulting from the input of Figure 1 and corresponding to the cut h_3. The arcs tickness represent the strength of the influence along that arc, i.e., the ticker the arc, the larger the associated probability.

exponential in the size of V. Moreover, using the structure of $\mathcal{H}$ is complicated by the possibly complex interplay between the likelihood and the model selection function f. Designing efficient algorithms might be possible, at least for some choices of f, but we leave those as future work.

Finally, it is worth noting that the two extreme cases outlined above, i.e., all links have the same probability, or all links have a different probability can be modeled in our framework. The cut h_1 in Figure 1 places all vertices of G in separate clusters, which corresponds to the most complex model with a separate influence probability on every edge. The cuts h_2 and h_3 induce models with a lower granularity. Finally, a cut right above the root of $\mathcal{H}$ (assuming that there is a "super-root" above r) places all

vertices in the same cluster, which results in the simplest possible model with a constant $p(u, v)$ for each arc (u, v).

3.3 Learning Inter-community Influence Strength

Next we devise an expectation-maximization (EM) approach for estimating the pairwise influence strength among the clusters of nodes, i.e., the parameters of the CSI model. We assume that the clusters have been induced by a cut of a given hierarchical decomposition $\mathcal{H}$ of G as discussed above, but the EM method presented in this section can be applied to an arbitrary disjoint partition of V. Recall that $a(v)$ denotes the cluster where $v \in V$, and let $A(x) \subseteq V$ denote the set of vertices that belong to cluster $x \in P$.

Consider a single trace $\alpha \in \mathbb{D}$. According to the discrete-time independent cascade model, each user u such that $u \in F^+_{\alpha,v}$ performs an independent attempt to activate v. If v becomes active then at least one of the influencers in $F^+_{\alpha,v}$ was successful, but we don't know which one. Hence, we introduce a probability distribution $\boldsymbol{\varphi}_{\alpha,v}$ over the nodes in $F^+_{\alpha,v}$, where $\varphi_{\alpha,v,u}$ represents the probability that in trace α the activation of v was due to the success of the activation trial performed by u.

We use these probabilities to derive a standard EM algorithm. For a given cut h, each $u \in F^+_{\alpha,v}$ succeeds in activating v on the considered trace with probability $p_h(a(u), a(v))$ and fails with probability $(1 - p_h(a(u), a(v)))$. By exploiting users' responsabilities $\varphi_{\alpha,v,u}$, we can define the complete expectation (log)likelihood of the observed propagation as follows:

$$Q(\tilde{p}_h; \tilde{p}_h^{\text{old}}) = \sum_{\alpha \in \mathbb{D}} \sum_{v} \left\{ \sum_{u \in F^+_{\alpha,v}} \Big(\varphi_{\alpha,v,u} \log \tilde{p}_h(a(u), a(v)) + (1 - \varphi_{\alpha,v,u}) \log \left(1 - \tilde{p}_h(a(u), a(v))\right) \Big) + \sum_{u \in F^-_{\alpha,v}} \log \left(1 - \tilde{p}_h(a(u), a(v))\right) \right\}. \tag{3}$$

Given an estimate of every $\varphi_{\alpha,v,u}$, we can determine the $\tilde{p}_h$ which maximizes Eq.3 by solving $\frac{\partial Q(\tilde{p}_h; \tilde{p}_h^{\text{old}})}{\partial \tilde{p}_h(x,y)} = 0$ for all pair of clusters $x, y \in P_h$. This gives the following estimate of $\tilde{p}_h(x, y)$:

$$\tilde{p}_h(x, y) = \frac{1}{|S^+_{x,y}| + |S^-_{x,y}|} \sum_{\alpha \in \mathbb{D}} \sum_{v \in A(y)} \sum_{\substack{u \in F^+_{\alpha,v} \\ u \in A(x)}} \varphi_{\alpha,v,u}, \tag{4}$$

where

$$S^+_{x,y} = \sum_{\alpha} \sum_{v \in A(y)} \sum_{u \in A(x)} \mathbb{I}(u \in F^+_{\alpha,v}) \text{ and } S^-_{x,y} = \sum_{\alpha} \sum_{v \in A(y)} \sum_{u \in A(x)} \mathbb{I}(u \in F^-_{\alpha,v}).$$

We still must provide an estimate for every $\varphi_{\alpha,v,u}$. We do this on the basis of the assumption that the probability distributions $\boldsymbol{\varphi}_{\alpha,v}$ are independent of the partition P. That

is, if u is believed to be the activator for v in the trace α, this belief should not change for different ways of clustering the two nodes. Therefore, we estimate the $\varphi_{\alpha,v,u}$s from the model where every $v \in V$ belongs to its own cluster, because this will lead to estimates that only depend on the network structure. Denote this model by $\tilde{p}_l$. We obtain:

$$\varphi_{\alpha,v,u} = \frac{\tilde{p}_l(v,u)}{1 - \prod_{w \in F^+_{\alpha,u}(1-\tilde{p}_l(w,u))}}. \tag{5}$$

Thus we design the following procedure:

- Run the EM algorithm without imposing a clustering structure (which is equivalent to the estimation proposed in [16]) to obtain $\tilde{p}_l(u,v) \forall (u,v) \in E$.
- Compute each $\varphi_{\alpha,v,u}$ using Equation 5.
- For different partitions P, keep the $\varphi_{\alpha,v,u}$ fixed, and update $\tilde{p}(x,y)$ according to Eq.4.

3.4 Model Selection

Recall that the likelihood $\log L(\mathbb{D} \mid p_h)$ is maximized for the cut h that places every node in its own cluster. We need thus a way to address the trade-off between fit and model complexity. Two principled approaches to this are *Bayesian Information Criterion* (BIC) and *Minimum Description Length* (MDL), which we both use in this paper.

We instantiate BIC [17] as follows: $\mathrm{BIC} = -2 \log L(\mathbb{D} \mid p_h) + |h| \log(|\mathbb{D}|)$.

In the basic two-part MDL [15], we first use the model, in our case the cut h, to *encode* the observed data (the traces in $\mathbb{D}$), and then encode the model itself. We denote the encoding length of $\mathbb{D}$ given h by $L(\mathbb{D} \mid h)$, and the encoding length of the cut by $L(h)$.

To apply MDL in our context, we must specify both $L(\mathbb{D} \mid h)$ as well as $L(h)$. A standard result is that we can simply use the log-likelihood of $\mathbb{D}$ given h as $L(\mathbb{D} \mid h)$ (see e.g. [3]). That is, we let $L(\mathbb{D} \mid h) = \log L(\mathbb{D} \mid p_h)$. The encoding length $L(h)$ of the cut h is defined as the number of bits needed to communicate h to a receiving party. We assume that the recipient already has the hierarchy $\mathcal{H}$ as well as the network G. We must send every edge in h using $|h| \cdot \log(|\mathcal{H}|)$ bits, as well as the influence probabilities between all pairs of communities where this probability is nonzero. If there are $X \leq |h|^2$ such pairs, we use $X\,(2\log(|h|) + C)$ bits (probabilities are encoded with C bits each) for the probabilities. The total encoding length of h is thus:

$$L(h) = |h| \cdot \log(|\mathcal{H}|) + X\,(2\log(|h|) + C)\,.$$

MDL favors the model that minimizes the combined encoding: $L(\mathbb{D} \mid h) + L(h)$.

4 Algorithm

In the previous section, we introduced the CSI model and discussed how to evaluate different cuts of $h \in \mathcal{C}(\mathcal{H})$ of the hierarchical decomposition of the network. However, as mentioned in Sec. 3, the search space $\mathcal{C}(\mathcal{H})$ can in general be exponential in the size

of $\mathcal{V}$, making exhaustive search infeasible. Next we present a heuristic algorithm that performs a *bottom-up* greedy visit of $\mathcal{C}(\mathcal{H})$, and provides the best solution found as output.

In our implementation, $\mathcal{H}$ is always a binary tree, but the approach applies to general trees as well. The procedure starts from the cut corresponding to the leaf level; at each iteration we compute all the possible cuts which can be obtained from the current one by merging communities that share a same parent in the hierarchy. Since $\mathcal{H}$ is a binary tree, each merge will involve exactly two communities. More formally, given a cut $h = \{e_1, \cdots, e_{|h|}\} \in \mathcal{C}(\mathcal{H})$, let $\mathcal{M}(h)$ denote the set of candidate merges available from the cut h:

$$\mathcal{M}(h) = \{\langle y, y\prime\rangle : (x, y) \in h \wedge (x, y\prime) \in h, x \neq r\}$$

where r is the root of $\mathcal{H}$. A simple greedy heuristic would pick the merge in $\mathcal{M}(h)$ that results in the best value of the objective function. However, evaluating our objective function is computationally intensive, because it involves re-estimating model parameters, and computing the likelihood of $\mathbb{D}$ given those parameters. This is too slow to be useful in practice.

To speed-up the algorithm, we make use of the following observation: in an "ideal merge" (with respect to the outgoing influence patterns) the two communities *exhibit exactly the same influence probabilities with other communities*. That is, if for some y and $y\prime$ we have $p(y, z) = p(y\prime, z)$ and $p(z, y) = p(z, y\prime)$ for every z, merging the communities y and $y\prime$ does not affect the likelihood of $\mathbb{D}$ at all. In practice these influence probabilities are never exactly identical, but we can still find a merge where they are *as similar as possible*. Rather than computing the entire objective function for every possible merge in $\mathcal{M}(h)$, we find the merge that is the best in terms of the above condition. To this end we use a similarity function defined as

$$\text{similarity}(y, y\prime, p) = \sum_z \left(p(y, z)p(y\prime, z) + p(z, y)p(z, y\prime)\right), \tag{6}$$

which can be thought of as the dot product between the influence probability vectors associated with communities y and $y\prime$.

Our whole procedure, summarized in Algorithm 1, finds in each iteration the best merge using Eq. 6 and updates the model given this. The resulting cut as well as the corresponding parameters are stored in a set, denoted L. Once the algorithm reaches the root of $\mathcal{H}$, it evaluates the objective function for every cut in the set L and returns the one having the best value. The function updateModel runs the estimation procedure described in Section 3.3.

5 Experimental Evaluation

CSI provides a compact description of influence patterns in the underlying network; in the following we will describe how this approach can be exploited for several purposes, including data understanding and characterization of information propagation flow.

Datasets. The evaluation focuses on three datasets where each dataset comprises of a network G and the propagation log $\mathbb{D}$. The first dataset has been extracted from Yahoo!

Algorithm 1. CSI model learning

Input : A propagation log $\mathbb{D}$, a network $G = (V, E)$ and hierarchical decomposition $\mathcal{H}$.
Output: The cut $h \in C(\mathcal{H})$ which achieves the best value of the objective function.
$h \leftarrow \text{leafLevel}(\mathcal{H})$
$p \leftarrow \text{updateModel}(h, G, \mathbb{D})$
$L \leftarrow \emptyset$
while $\mathcal{C}(h) \neq \emptyset$ **do**
  $\langle x^*, y^* \rangle \leftarrow \arg\max_{\langle x,y \rangle \in \mathcal{C}(h)} \{\text{similarity}(x, y, p)\}$
  $h \leftarrow \text{merge}(h, \langle x^*, y^* \rangle)$
  $p \leftarrow \text{updateModel}(h, G, \mathbb{D})$
  $L \leftarrow L \cup \{\langle h, p \rangle\}$
end
$\langle h^*, p^* \rangle \leftarrow \arg\min_{\langle h,p \rangle \in L} \{\text{objFunc}(h, p, \mathbb{D})\}$
return h^*

Table 1. Datasets statistics

	MEME	FLIXSTER	TWITTER
Number of Nodes	9,523	6,354	23,537
Number Links	759,369	97,314	1,299,652
Traces	9,578	7,158	6,139
Activations	552,732	1,439,875	383,866
Avg. number of activations per node	84	221	16
Avg. number of nodes per trace	58	200	62

Meme, a microblogging service[1], in which users can share different kinds of information called "memes". Memes are shared on the main user's stream and a re-post button allows to display an item from another user's stream on the personal one. If the user v posts a meme which is later re-posted by the user u, we say that the meme propagates from v to u, and thus v is a potential influencer of u. The second dataset has been crawled from Flixster[2], one of the main social movie website. It allows users to share ratings on movies and to meet other users with similar tastes. In Flixster,the propagation log records the time at which a user rated a particular movie. In this context, an item or movie is considered to propagate from v to u, if u rates the item shortly after the rating by v. The last dataset was obtained by crawling the public timeline of Twitter[3]. We track the propagation of URLs across the network where an activation corresponds to the instance when a user uses a certain URL for sharing with other friends. In Table 1 we report the main characteristics of the datasets.

Experiment Settings. The optimization algorithm proposed in Sec. 4 requires as input a hierarchical decomposition of the network. We obtain this hierarchy by recursively partitioning the underlying network using *METIS* [9], which reportedly provides high

[1] Discontinued in May 25, 2012.
[2] `http://www.cs.sfu.ca/~sja25/personal/datasets/`
[3] `https://dev.twitter.com/docs/api/1/get/statuses/public_timeline`

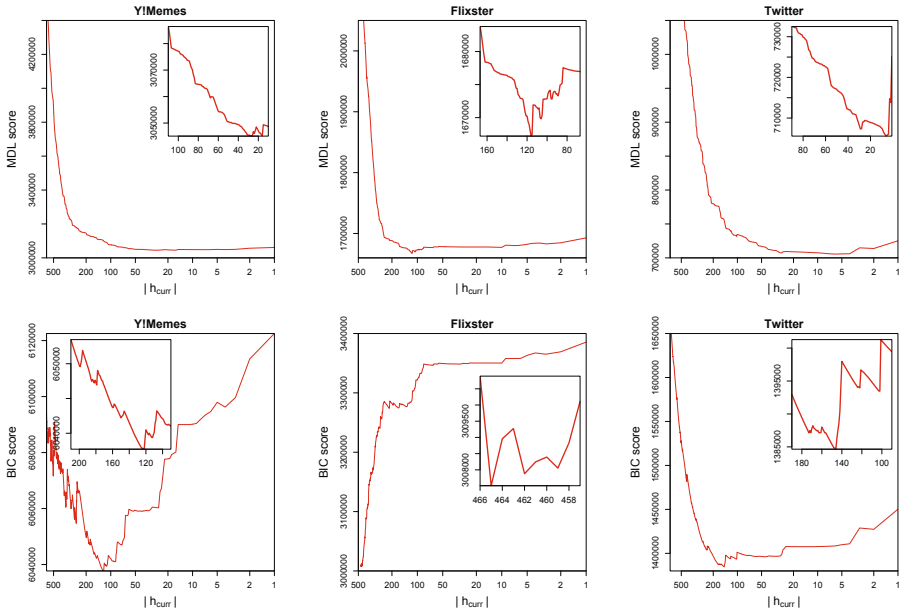


Fig. 3. Model selection: MDL (first row), and BIC (second row) - the subplots show a focused view on the region of the minimum

quality partitions. At each stage in the recursive procedure, we split the network to two components that are roughly equal in size. The resulting hierarchical partitioning is thus a binary tree. In order to reduce the computational overhead of the algorithm, we initialize the hill climbing clustering procedure with a cut slightly above the leaf level. This is obtained by making 5 passes over the leafs, and during each pass we merge leafs that have a common parent. While the choice of the similarity function is somewhat arbitrary, we found that the cosine similarity of Eq. 6 works well in practice. Finally, the delay threshold Δ is set to ∞: i.e., for a given node we consider potential influencer any neighbor active before the node in a given trace. We ran our experiments on a Intel Xeon 2.4 GHz processor and 8 GB memory. The learning time ranges from few hours (Flixster and Y!Meme) to several days for Twitter, where the number of links (approx. 1.3 million) impact the learning time as it increases the computational effort in Eq. 4 where a greater number of potential activators of v needs to be considered. It is worth noting that parallelizing the EM computation of Section 3.3 is possible and it is planned in our future work.

Model Selection. In Figure 3, we compare the BIC and MDL scores that we obtain for each cut found by our algorithm from the lowest (many communities) to the highest (few communities). The two model selection criteria do not agree on the identification of the optimal model. MDL tends to favor less complex models than BIC in our case, which is most likely caused by the quadratic dependence on $|h|$ of $L(h)$. For instance, MDL provides us 115 communities for Flixster dataset whereas BIC provides 454 communities (just after a couple of iterations of the main algorithm). In the rest of

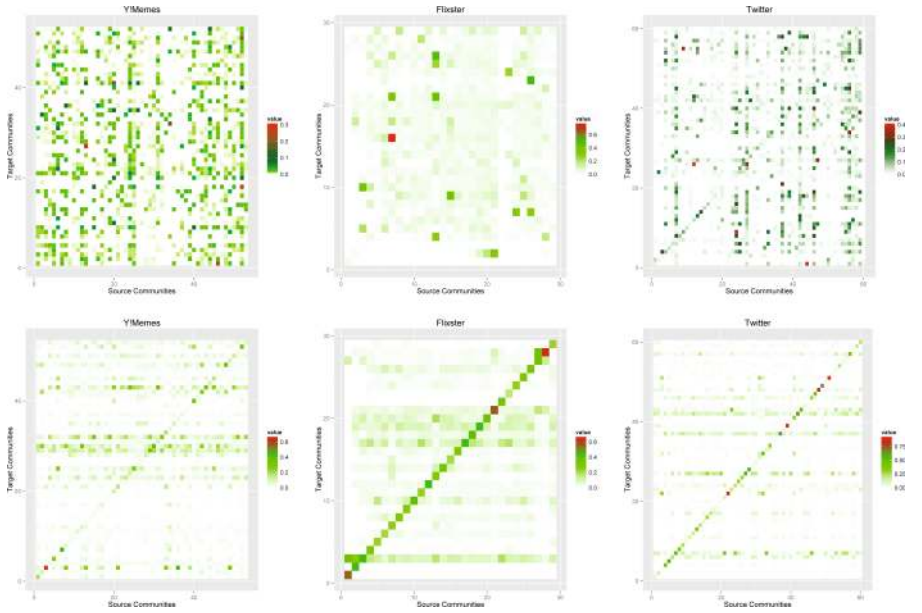


Fig. 4. Community to community influence probabilities (first row) and social links (second row)

this section we will characterize one model found for each dataset. More specifically, for Flixster we select the model provided by MDL since it provides a more compact view of the influence pattern, with (29 communities, after removing singleton node communities). For the other two datasets, we select the model provided by BIC. The number of communities for Twitter and Y!Meme, after removing singletons, are 60 and 53, respectively.

Community-Level Social Influence Analysis. The output of the CSI model can be easily graphically represented according to two different, and complementary, perspectives. The first way to analyze the strength of the influence and social link relationships is by plotting the corresponding heat-maps, as shown in Figure 4. In these figures, we plot the intensity of the influence probability between two communities, and the probability of observing a link between them, respectively. On the whole, we register almost no correlation between influence and link probabilities. From the heat maps corresponding to link probabilities, we can see that the clustering procedure use to find the hierarchy $\mathcal{H}$ (METIS) has correctly identified communities of highly connected nodes. Influence relationships, however, do not in general exhibit any clear structure, although we have a slight diagonal in the Twitter dataset. Interestingly, even if a community is dense, it does not necessarily exhibit strong internal influence.

An alternate and perhaps more effective way of summarizing influence relationships in the network is to consider the community-level influence propagation network. In Figure 5 we show the CSI propagation network for the Flixster dataset, where node

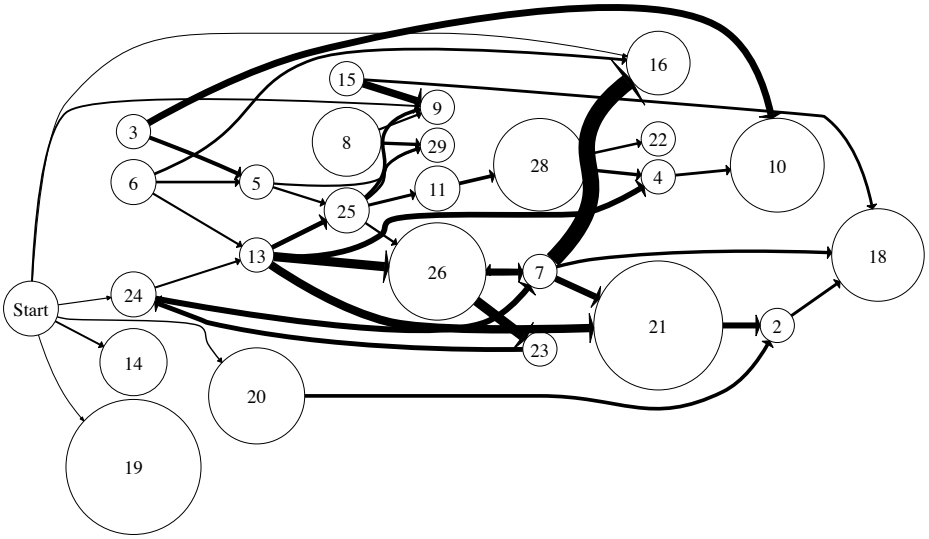


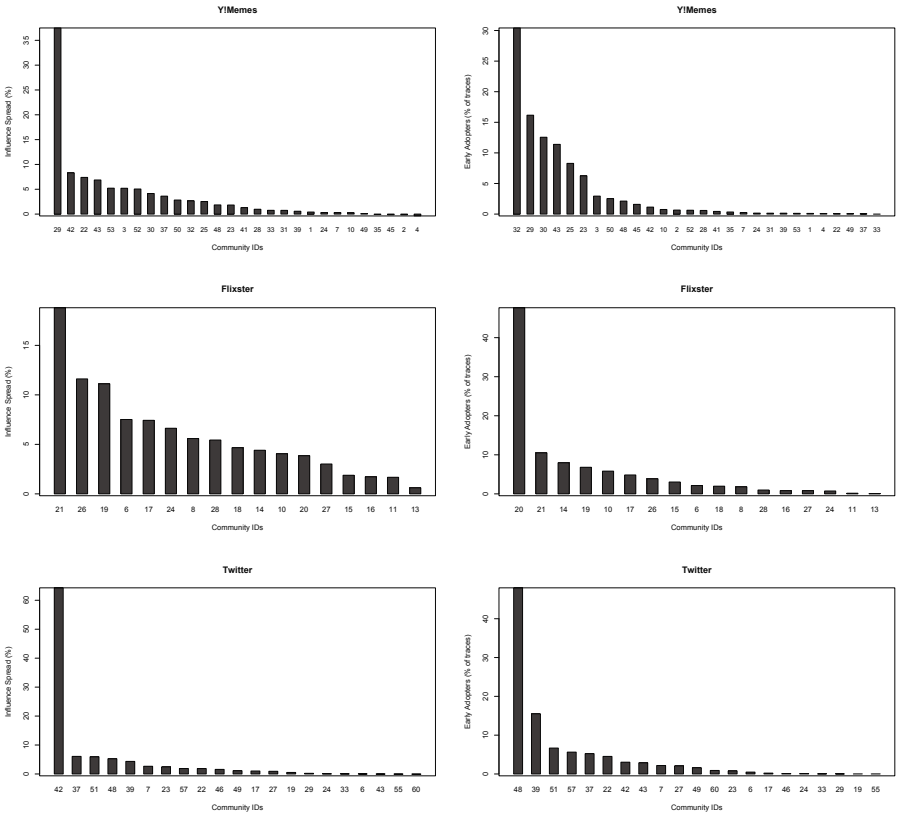
Fig. 5. The CSI model found in the Flixster dataset

size is proportional to community size, "Start" represents the Ω node, and edge width is proportional to influence probability. We preprocessed the graph by pruning all the edges between communities having influence probability less than 0.1, while we use 0.03 as threshold for the links connecting Ω with the rest of the network. Interestingly, the network is almost acyclic, and this suggests a clear directionality pattern in the flow of information. This finding is further confirmed by the analysis of the other two datasets, for which, due to the limited space, we omit the CSI propagation networks.

Both the heat-map representation and the compact propagation network provide an useful tool to understand influence relationships between communities, and at the level of the whole network. We can evaluate the capabilities of this approach in providing a compact and yet accurate description of the real influence process on the underlying network, by setting up two simple tests.

In the first test, the goal is to verify if our approach detects correctly the communities that play a key role in the information propagation process. To quantify the importance of each community, we run the greedy influence maximization algorithm [10] (with a budget of 100 nodes) on the considered networks, where the influence probabilities are the ones estimated at the leaf level and we employ 5000 Monte Carlo simulation to estimate the spread. This procedure provides a ranked list $\mathcal{S}$ of nodes that should be targeted to maximize the expected spread on the network. For each $s \in \mathcal{S}$ we record Δ_s as the gain, in terms of spread, that can be achieved by adding s to the set $\mathcal{S}$. We can measure its importance in the overall diffusion process as the percentage of the overall spread achieved by targeting the seed nodes selected in the considered community. More formally:

$$\text{score}(c) = \frac{\sum_{s' \in \mathcal{S} \wedge s' \in A(c)} \Delta_{s'}}{\sum_{s' \in \mathcal{S}} \Delta_{s'}}.$$



(a) Influence spread score (b) Early adoption score

Fig. 6. Influence spread and early adoption scores of the communities produced by CSI

As clearly visible in Figure 6(a), a high fraction of the overall spread can be explained by only a few communities. The high score of the most important communities is due to the fact that the multiple nodes having a large spread gain belong to those communities. As instance, the greedy procedure picks 13 nodes in the community 29 for Meme, 17 belonging to the community 21 for Flixster, 13 in the community 42 for Twitter. Interestingly, the community structure provided by our approach gathers "high influential" nodes in the same community.

The second validation test to provide a qualitative evaluation of the CSI model is focused on the identification of early adopter communities. For each trace we consider the communities that have highest number of active nodes during the first quarter of the trace's overall propagation time, and rank communities accordingly. Again, as shown in Figure 6(b), a significant number of traces involves the initial activation of nodes belonging to a small set of communities. The identity of those communities can be easily tracked by considering the length of the path from the "Start" node in the CSI propagation graph. As instance, on Flixster (Figure 5), the communities 14 and 19 exhibit

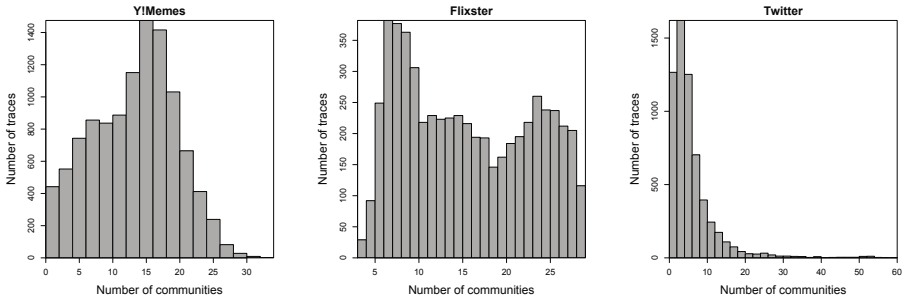


Fig. 7. Distribution of number of communities touched by propagation traces

a direct connection with the "Start" node, while community 21, can be reached in two hops, characterized by an high influence weight.

Finally, the community structure detected by CSI can be useful to study the properties of the information propagation flow on the considered networks. As instance, we may be interested in studying the typical flow of information in the network, by analyzing the number of communities reached. We provide, in Figure 7, the histogram of the number of communities reached by the information propagation traces, where we consider that a trace enters in a community if at least 10% of its nodes become active. We observe different information propagation patterns: in Flixster a larger fraction of traces propagate in a small number of communities. The number of communities participating in a propagation declines as the number of communities increase, however, it rises again for relatively large number of communities and, thereby, making a near U-shaped distribution. Hence, a significant number of traces exhibit either local or global diffusion. In Y!Meme, the number of communities involved in the propagation flow follows a normal distribution, and again only a limited number of communities (15 in average) are typical involved in the propagation. This "local propagation" behavior is emphasized on Twitter, where each trace generally involves less than 5 communities.

6 Conclusions

In this paper we introduce a hierarchical approach to summarize patterns of influence in a network, by detecting communities and their reciprocal influence strength. Our model, dubbed CSI, generalizes the Independent Cascade propagation model, by modeling influence between communities of connected nodes rather than a pairwise node influence. This enables more compact representation of the network of influence, which can be easily plotted and exploited to understand and detect interesting properties in the information propagation flow over the network. Our empirical analysis over real-world networks highlights two interesting observations: (i) the propagation networks found by CSI are almost acyclic, (ii) information propagates in the network mainly "locally", reaching few communities. While the first observation offers interesting insights, since it shows the existence of a clear direction in the propagation of information, the latter confirms a strong relationships between information propagation and the community structure, that might be exploited for community detection [1].

Acknowledgments. This research was partially supported by the Torres Quevedo Program of the Spanish Ministry of Science and Innovation, and partially funded by the European Union 7th Framework Programme (FP7/2007-2013) under grant n. 270239 (ARCOMEM).

References

1. Barbieri, N., Bonchi, F., Manco, G.: Cascade-based community detection. In: WSDM 2013 (2013)
2. Bonchi, F., Castillo, C., Donato, D., Gionis, A.: Taxonomy-driven lumping for sequence mining. Data Mining and Knowledge Discovery 19(2), 227–244 (2009)
3. Cover, T.M., Thomas, J.A.: Elements of information theory. Wiley-interscience (2012)
4. Domingos, P., Richardson, M.: Mining the network value of customers. In: KDD 2001 (2001)
5. Fortunato, S.: Community detection in graphs. Physics Reports 486(3), 75–174 (2010)
6. Garriga, G.C., Ukkonen, A., Mannila, H.: Feature selection in taxonomies with applications to paleontology. In: Boulicaut, J.-F., Berthold, M.R., Horváth, T. (eds.) DS 2008. LNCS (LNAI), vol. 5255, pp. 112–123. Springer, Heidelberg (2008)
7. Goyal, A., Bonchi, F., Lakshmanan, L.V.S.: Learning influence probabilities in social networks. In: WSDM 2010 (2010)
8. Goyal, A., Bonchi, F., Lakshmanan, L.V.S.: A data-based approach to social influence maximization. PVLDB 5(1), 73–84 (2011)
9. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J. Sci. Comput. 20(1), 359–392 (1998)
10. Kempe, D., Kleinberg, J.M., Tardos, É.: Maximizing the spread of influence through a social network. In: KDD 2003 (2003)
11. Lavrač, N., Vavpetič, A., Soldatova, L., Trajkovski, I., Novak, P.K.: Using ontologies in semantic data mining with segs and g-segs. In: Elomaa, T., Hollmén, J., Mannila, H. (eds.) DS 2011. LNCS, vol. 6926, pp. 165–178. Springer, Heidelberg (2011)
12. Mathioudakis, M., Bonchi, F., Castillo, C., Gionis, A., Ukkonen, A.: Sparsification of influence networks. In: KDD 2011 (2011)
13. Navlakha, S., Rastogi, R., Shrivastava, N.: Graph summarization with bounded error. In: SIGMOD 2008 (2008)
14. Navlakha, S., Schatz, M.C., Kingsford, C.: Revealing biological modules via graph summarization. Journal of Computational Biology 16(2), 253–264 (2009)
15. Rissanen, J.: A universal prior for integers and estimation by minimum description length. The Annals of Statistics, 416–431 (1983)
16. Saito, K., Nakano, R., Kimura, M.: Prediction of information diffusion probabilities for independent cascade model. In: Lovrek, I., Howlett, R.J., Jain, L.C. (eds.) KES 2008, Part III. LNCS (LNAI), vol. 5179, pp. 67–75. Springer, Heidelberg (2008)
17. Schwarz, G.: Estimating the dimension of a model. The Annals of Statistics 6(2), 461–464 (1978)
18. Tang, J., Sun, J., Wang, C., Yang, Z.: Social influence analysis in large-scale networks. In: KDD 2009 (2009)
19. Tian, Y., Hankins, R.A., Patel, J.M.: Efficient aggregation for graph summarization. In: SIGMOD 2008 (2008)
20. Xiang, R., Neville, J., Rogati, M.: Modeling relationship strength in online social networks. In: WWW 2010 (2010)
21. Zhang, N., Tian, Y., Patel, J.M.: Discovery-driven graph summarization. In: ICDE 2010 (2010)