# Communities of Interest

Corinna Cortes, Daryl Pregibon & Chris Volinsky

AT&T Shannon Research Labs
Florham Park, New Jersey, USA

**Abstract.** We consider problems that can be characterized by large dynamic graphs. Communication networks provide the prototypical example of such problems where nodes in the graph are network IDs and the edges represent communication between pairs of network IDs. In such graphs, nodes and edges appear and disappear through time so that methods that apply to static graphs are not sufficient. We introduce a data structure that captures, in an approximate sense, the graph and its evolution through time. The data structure arises from a bottom-up representation of the large graph as the union of small subgraphs, called Communities of Interest (COI), centered on every node. These subgraphs are interesting in their own right and we discuss two applications in the area of telecommunications fraud detection to help motivate the ideas.

## 1 Introduction

Transactional data consists of records of interactions between pairs of entities occurring over time. For example, a sequence of credit card transactions consists of purchases of retail goods by individual consumers from individual merchants. Transactional data can be represented by a graph where the nodes represent the transactors and the edges represent the interactions between pairs of transactors. Viewed in this way, interesting new questions can be posed concerning the connectivity of nodes, the presence of atomic subgraphs, or whether the graph structure leads to the identification and characterization of "interesting" nodes. For example, Kleinberg (1998) introduces the notion of "hubs" and "authorities" as interesting nodes on the internet. The data used by Kleinberg differ significantly from the data we consider in that he uses static links to induce a graph over web pages. In our case, we use actual network traffic, as captured by interactions between pairs of transactors, to define our graph. Thus in a very real sense, the graph we consider is dynamic since nodes and edges appear and disappear from the graph through time.

There are many challenging issues that arise for dynamic graphs and we have used a specific application to focus our research, namely the graph induced by calls carried on a large telecommunications network. This application is interesting, both because of its size (i.e., hundreds of millions of nodes) and its rate of change (i.e., hundreds of millions of new edges each day). Like all networks, it is also diverse in the sense that some nodes are relatively inactive while others are superactive.

In thinking about dynamic graphs, the first question that arises concerns the definition of $\mathcal{G}_t$, namely the graph $\mathcal{G}$ at time $t$. The intuitive notion is that $\mathcal{G}_t$ consists of the nodes and edges active at time $t$, or in a small interval around $t$. We consider discrete time applications where new sets of nodes and edges corresponding to the transactions from time step $t$ to $t+1$ only become available at the end of the time step, for example once a day. Associated with every edge is a weight that is derived from an aggregation function applied to all (directed) transactions between a pair of nodes at time step $t$. For example, the aggregation function can be the "total duration of calls" or the "number of calls" from one node to another.

Let the graph corresponding to the transactions during time step $t$ be $g_t$. We can define $\mathcal{G}_t$ from $g_i$ where $i = 1, \ldots, t$ in several ways. Let us first define the sum of two graphs $g$ and $h$

$$G = \alpha g \oplus \beta h$$

where $\alpha$ and $\beta$ are scalars. The nodes and edges in $G$ are obtained from the union of the nodes and edges in $g$ and $h$. The weight of an edge in $G$ is

$$\mathrm{w}(G) = \alpha \mathrm{w}(g) + \beta \mathrm{w}(h)$$

where the weight of an edge is set to zero if the edge is absent from the graph.

If one defines $\mathcal{G}_t = g_t$, $\mathcal{G}_t$ is very unstable as it might change dramatically at each time step. On the other hand, if one defines

$$\mathcal{G}_t = g_1 \oplus g_2 \oplus \ldots \oplus g_t = \bigoplus_{i=1}^{t} g_i,$$

$\mathcal{G}_t$ is perhaps too stable, as it includes all historic transactions from the beginning of time. To allow the graph $\mathcal{G}_t$ to track the dynamics of the transactional data stream, one can define $\mathcal{G}_t$ as a moving window over $n$ time steps:

$$\mathcal{G}_t = g_{t-n} \oplus g_{t-n+1} \oplus \ldots \oplus g_t = \bigoplus_{i=t-n}^{t} g_i.$$

However, this definition suffers from two problems: first, one needs to store the graphs $g_i$ corresponding to the last $n$ time steps to compute $\mathcal{G}_t$, and second, it gives equal weight to all time steps.

To allow for a smooth dynamic evolution of $\mathcal{G}_t$ without incurring the storage problems of the moving window approach, we adopt the recursive definition:

$$\mathcal{G}_t = \theta \mathcal{G}_{t-1} \oplus (1 - \theta) g_t \tag{1}$$

where $0 \leq \theta \leq 1$ is a parameter that allows more ($\theta$ near 1) or less ($\theta$ near 0) history to influence the current graph. An alternative representation of (1) is obtained by expanding the recursion:

$$\mathcal{G}_t = \omega_1 g_1 \oplus \omega_2 g_2 \oplus \ldots \oplus \omega_t g_t = \bigoplus_{i=1}^{t} \omega_i g_i \tag{2}$$

where $\omega_i = \theta^{t-i}(1-\theta)$. This representation highlights the fact that more recent data contributes more heavily to $\mathcal{G}_t$ than older data. Figure 1 displays this graphically. If processing occurs daily, then a value of $\theta = 0.85$ roughly corresponds to $\mathcal{G}_t$ capturing a moving month of network activity. Finally note that the convex combination defined in (1) is "smoother" than a simple 30 day moving window as network peaks and troughs, induced say by holiday traffic patterns, are effectively moderated by $\theta$.

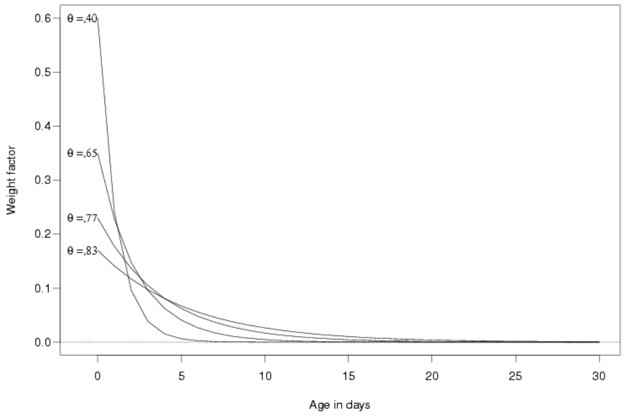


**Fig. 1.** *Damping factor of edge weights as a function of time steps (days) in the recursive definition (2) of $\mathcal{G}_t$. The values of $\theta$ correspond to effectively no influence from an edge after 1 ($\theta = 0.40$), 2 ($\theta = 0.65$), 3 ($\theta = 0.77$), or 4 weeks ($\theta = 0.83$).*

The paper is organized as follows. Section 2 describes the data structure that we use to capture network activity and to evolve it through time. We also briefly discuss the strategy we employ to traverse this data structure to quickly and efficiently build graphs around individual nodes. Section 3 introduces a pair of examples that illustrate how these subgraphs are used in practice. Section 4 summarizes the findings and discusses future work.

## 2 Data Structure

We propose a constructive approach to evolving a large time-varying graph. Consider a node in the graph, its associated directed edges, and weights associated with each edge. A data structure that consists of these weighted directed edge sets for each node is a representation of the complete graph. This data structure is redundant since it is indexed by nodes so that edges must be stored twice, once for the originating node and once for the terminating node. In contrast, a data structure that stores each edge once must be doubly indexed by nodes. The cost of edge duplication is often mitigated by gains in processing speed when subgraphs around nodes are expanded. For this reason we have chosen to represent our graphs as a singly indexed list of nodes, each with an associated array of weighted directed edges.

The data structure outlined above is complete in the sense that it captures the entire graph. However in the applications that we are familiar with, the computational horsepower to maintain complete graphs with hundreds of millions and nodes and billions of edges is neither feasible nor desirable. Instead we define a new graph where the atomic unit is the subgraph consisting of a node and its directed top-$k$ edges to other nodes. The meaning of "top" is relative to the aggregation function applied to transactions associated with each edge, so it might be the top-$k$ edges in terms of the "number of calls." In addition to the top-$k$ inbound and top-$k$ outbound edges, we also define an overflow node, called "other", for aggregating traffic to/from nodes not contained in the top-$k$ slots. While the value of $k$ determines how well our data structure approximates the true network graph, it is worth noting that larger is not necessarily better when network graphs are used to study connectivity of the nodes. For example, assuming that a few percent of all calls are misdialed, do we really want the corresponding edges reflected in the graph? In our experience we have found the answer to this question to be "no" and have used a value of $k$ that balances computational complexity (e.g., as regards speed and storage) with empirically determined accuracy (see below).

In practice, the edges may not always be retained in a symmetric fashion. If one node receives calls from many nodes, like 800CALLATT, it overflows its top-$k$ slots so most callers will find their edges absorbed by the "other" node. However, any single node making calls to 800CALLATT may not have edges to more than $k$ other nodes, and the edge will be contained in its top-$k$ edge set. Duplication of edges becomes complete only in the limit as $k \to \infty$. For all finite $k$, the only invariance over the set of subgraphs is that the sum of the outbound edge weights equals the sum of the inbound edge weights, where the sum includes node "other". For most of our applications, we use $k = 9$ since most residential long distance accounts do not exceed this number of edges (see Figure 2).

To accommodate the time evolution of the network graph, the data structures are updated at fixed time steps. Between updating steps, transactions are collected and temporarily stored. At the end of that time period, the transactions are aggregated and the subgraph updated. The length of the time period represents another trade-off in accuracy: the longer the time period, the better an estimate of the top-$k$ edge set, but the more outdated the resulting subgraph. In the applications discussed in Section 3, we perform daily updates, thereby maintaining reasonable accuracy while requiring temporary disk space for only one day of data. For a more detailed discussion see Cortes and Pregibon (1999).

Let $\hat{\mathcal{G}}_{t-1}$ denote the top-$k$ approximation to $\mathcal{G}_{t-1}$ at time $t-1$ and let $g_t$ denote the graph derived from the new transactions at time step $t$. The approximation to $\mathcal{G}_t$ is formed from $\hat{\mathcal{G}}_{t-1}$ and $g_t$, node by node, using a top-$k$ approximation to Eq. 1:

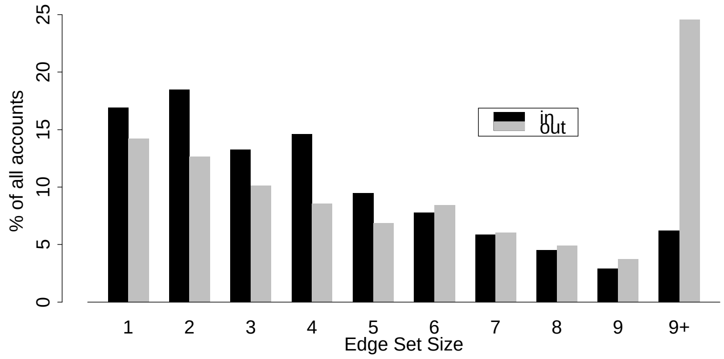


**Fig. 2.** *Edge set sizes for a random sample of residential accounts. Black bars indicate the size of the inbound edge set and gray bars indicate the size of the outbound edge set.*

$$\hat{\mathcal{G}}_t = \text{top-}k\{\theta\hat{\mathcal{G}}_{t-1} \oplus (1-\theta)g_t\} \tag{3}$$

Thus, we first calculate the edge weights for all the edges of $\theta\hat{\mathcal{G}}_{t-1} \oplus (1-\theta)g_t$. The overflow node "other" is treated as any other node in the graph. Then for each node we sort the edges according to their weight. The top-$k$ are preserved, and if there are more than $k$ edges in the edge set for that node, the weights of the remaining edges are added to the weight of the edge going from the node to node "other". These operations are displayed pictorially in Figure 3 using $\theta = .85$.

$$\begin{array}{cccccc}
 & \begin{array}{c}\text{Old top-}k\text{ edges}\\ \text{node–labels} \quad \text{wts}\end{array} & & \begin{array}{c}\text{Today's edges}\\ \text{node–labels} \quad \text{wts}\end{array} & & \begin{array}{c}\text{New top-}k\text{ edges}\\ \text{node–labels} \quad \text{wts}\end{array} \\
\theta & \begin{pmatrix} XXX6525467 & 5.2 \\ XXX7562656 & 5.0 \\ XXX6524132 & 4.5 \\ XXX6534231 & 2.3 \\ XXX6243142 & 1.9 \\ XXX7354212 & 1.8 \\ XXX4231423 & 0.8 \\ XXX5342312 & 0.5 \\ XXX5264532 & 0.2 \\ \text{Other} & 0.1 \end{pmatrix} & +\,(1-\theta) & \begin{pmatrix} XXX6525467 & 2.0 \\ XXX7562656 & 6.2 \\ XXX6524132 & 0.8 \\ XXX5436547 & 10.0 \\ & \\ & \\ & \\ & \\ & \\ \text{Other} & 0.0 \end{pmatrix} & = & \begin{pmatrix} XXX7562656 & 5.2 \\ XXX6525467 & 4.6 \\ XXX6524132 & 3.9 \\ XXX6534231 & 2.0 \\ XXX6243142 & 1.6 \\ XXX7354212 & 1.5 \\ XXX5436547 & 1.5 \\ XXX4231423 & 0.7 \\ XXX5342312 & 0.4 \\ \text{Other} & 0.3 \end{pmatrix}
\end{array}$$

**Fig. 3.** *Computing a new top-k edge set from the old top-k edge set and today's edges. Note how a new edge enters the top-k edge set, forcing an old edge to be added to Other.*

The subgraph consisting of the top-$k$ inbound and the top-$k$ outbound edges of a node is ideal for fast extraction of larger subgraphs centered on the node. The data structures can be queried recursively for each node in the top-$k$ edge sets of the center node. We grow such subgraphs in a breadth-first traversal of the data structure. For notational purposes, we denote the edge set of the node itself by $d_1$ (depth of 1). Similarly let $d_2$ denote the edge set formed by the union of $d_1$ and the edge sets of all nodes contained in $d_1$. We rarely explore edge sets greater than $d_2$ in our applications as they become unmanageably large and remarkably uninformative. Indeed we often apply a thresholding function to the edge weights, even for $d_2$, to further reduce "clutter" in a subgraph. Thus any edge with weight less than $\epsilon$ need not be expanded if one feels that such edges are inconsequential for the application at hand.

In the next section we introduce two applications that exploit the index structure of our representation. This is critical since we often need to compute and compare many subgraphs on a daily basis. We have tuned our algorithms so that the average time for retrieving and rendering a $d_2$ edge set from our data structure of close to 400M nodes is just under one second (on a single processor).

## 3 Applications

In the telecommunications industry, there are many different types of fraudulent behavior. Subscription fraud is a type of fraud that occurs when an account is set up by an individual who has no intention of paying any bills. The enabler in such cases involves either flawed processes for accepting and verifying customer supplied information, or identity-theft where an individual impersonates another person. In either case, if left undetected, the fraud is typically only discovered when the bill is returned to sender, often after thousands of dollars have been lost. In the first example we use COI and a "guilt by association" argument to detect new cases of fraud in the network. The second example uses a distance metric between COI to suggest that a fraudster has assumed a new network identity.

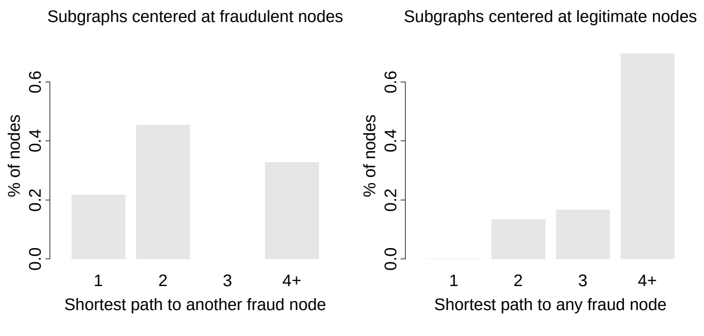


**Fig. 4.** *Guilt by association - what is the shortest path to a fraudulent node?*

## 3.1 Guilt by Association

When a new account is activated on the network, is there any way to label it according to its inherent riskiness? In this section we explore this possibility by defining a procedure that assesses risk on the basis of a node's connectivity to other nodes. To illustrate the idea we use data on subscribers from a large metropolitan city for a one month period. We have labeled the nodes as fraudulent or legitimate according to the disposition determined by network security.

Figure 4 displays the distribution of the number of edges from a node to the closest node that is labeled as fraudulent. The figure shows that fraudsters tend to be closer to other fraudsters than random accounts are to fraud. Specifically we see that relatively few legitimate accounts are directly adjacent to fraudulent accounts. Indeed network security investigators have found that fraudsters seldom work in isolation from each other. There are brokers who compromise a legit customer's service and then sell this service to their own set of customers. Or even if there is no organized criminal element operating, fraudsters often cannot stop themselves from sharing their tricks with their friends and family. Because fraudsters seem to form an affinity group, we label the network connectivity process to catch fraud as "guilt by association."

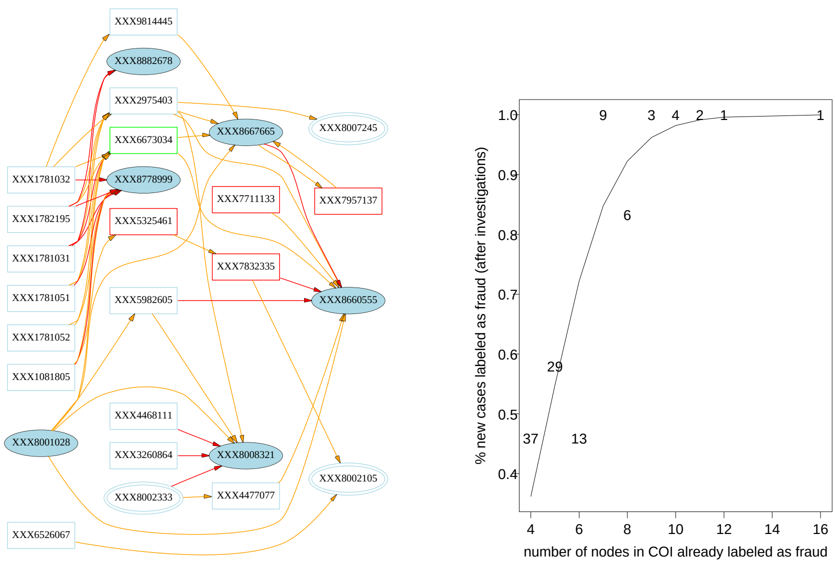


**Fig. 5.** Left Panel. *A guilt by association plot. Circular nodes correspond to wireless service accounts while rectangular nodes are conventional land line accounts. Shaded nodes have been previously labeled as fraudulent by network security associates.* Right Panel. *Calibration plot. The per cent fraud is plotted against the number of fraudsters in a COI for 105 cases presented to security. The plotting symbol is the number of these cases at each level of fraud infection. The curve superimposed on the points is the fit of a simple logistic model.*

Our process consists of the following steps:

- compute the $d_2$ edge sets for all new accounts one week after they are activated on the network
- label each of the nodes in the resulting COI as fraudulent or legitimate based on the most recent information from security associates
- rank the new accounts according to how much fraud appears in their COI

The left panel in Figure 5 provides an illustrative example where five nodes surrounding a new suspect (labeled XXX8667665) were recently deactivated for fraudulent behavior. The right panel summarizes the performance of the methodology for 105 cases presented to network security. While there is noise in the plot, it clearly shows that the probability that an account is fraudulent is an increasing function of the number of fraudulent nodes in its COI.

### 3.2 Record Linkage Using COI-based Matching

Consider the case where we have information on an account that was recently disconnected for fraud and we are looking for a new account that has the same individual behind it. Assuming that identity-theft was the root cause of the prior fraudulent account, it is likely that the new account is under a different name and address than the old one (i.e., the fraudster has now assumed the identity of a new victim). We attack this problem with the intuition that while the subscription information is not useful for matching network IDs to the same individual, the calling patterns of the new account, as characterized by its COI, should not change very much from the previous account. The left panel of Figure 6 shows a convincing case where two nodes appear to belong to the same individual. We now have a problem of matching COI, with the underlying problem of deriving a reasonable distance function to quantify the closeness of a pair of COI.

The matching problem is computationally difficult because of the size of our network – each day we see tens of thousands of new accounts. For each of these, we need to compute their COI, and then the distance from each of these to the COI of all recently confirmed fraudulent accounts. Assuming for these purposes that we maintain a library of the most recent 1000 fraudulent accounts, tens of millions of pairwise distances need to be computed. To carry out the computations we use a $d_2$ COI for all accounts in our "fraud library" and $d_1$ COI for all new accounts.

The distance between two COI depends on both the quantity and the quality of the overlapping nodes. The quantity of the overlap is measured by counting the number of overlapping nodes and calculating the percentage of a COI which consists of overlapping nodes. However all overlapping nodes are not equally informative, so we need a measure of quality as well. Many graphs will intersect at high-use nodes, such as large telemarketing shops or widely advertised customer service numbers. An informative overlapping node is one that has relatively low in- and out-degree, and in the best case, is shared only by the nodes under consideration for a match. We now describe a measure that captures these notions.

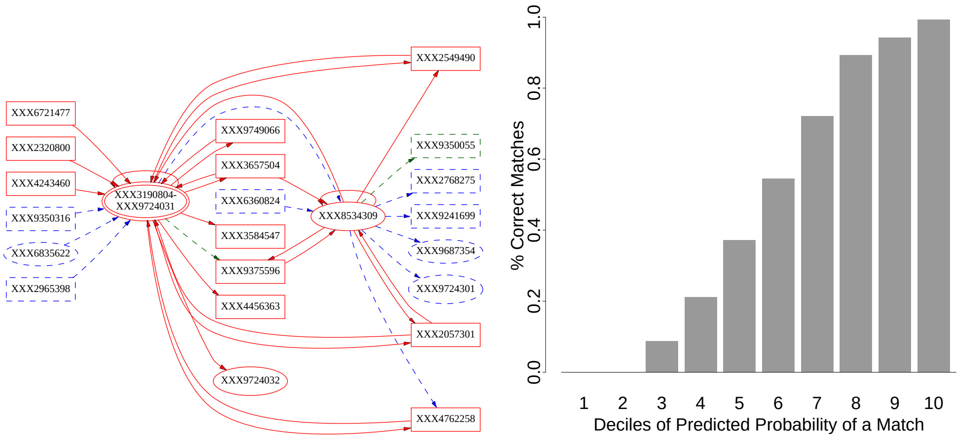


**Fig. 6.** Left Panel. *Visualization of linking accounts by their COI. The two individual COI are superimposed (in the double-lined oval) to show their similarity. Solid lines indicate edges common to both COI, while dashed lines and nodes indicate subgraphs belonging to only one COI.* Right Panel. *Success of COI matching. Observed proportions of matching node-pairs versus decile of predicted matching probability.*

Given two COI for nodes $a$ and $b$ with a non-zero overlap, we define

$$\text{Overlap}(\text{COI}_a, \text{COI}_b) = \sum_{o \in O} \frac{w_{ao} w_{bo}}{w_o} \frac{1}{d_{ao}} \frac{1}{d_{bo}},$$

where $O$ is the set of all overlapping nodes in the two COI, $w_{ao}$ is the weight of edges between node $a$ and node $o$ in $a$'s $d_1$ edge set, $w_o$ is the overall weight of node $o$ (the sum of all edges in the $d_1$ edge set for node $o$), and $d_{ao}$ is the minimal distance from node $a$ to node $o$ in $\text{COI}_a$. [In the case where $d_{ao} > 1$, it is not clear what the weight $w_{ao}$ should be, since there is no direct edge between the two. For this application we elected to set such a weight at $w_{ao} = .01$ in order to minimize the effect of these overlaps]. Intuitively, the numerator measures the strength of the connection from $a$ and $b$ to the overlap, while the denominator corrects for an overlap node which is either common to many nodes or is further in the graph from $a$ or $b$. This measure scores high for overlap nodes that have strong links to the nodes of interest, but otherwise have low overall volume.

A decision tree built with this score, along with several covariates obtained from the information provided by the subscriber, produces a "matching" probability for any list of node pairs. For some of the node pairs that we scored, investigators were able to determine whether the old and new accounts belonged to the same individual. The right panel of Figure 6 shows the performance of our COI matching model. For the sample of pairs that we validated ($n = 1537$), we display the observed proportion of matching node-pairs for each decile of predicted matching probability.

## 4 Conclusions

In this paper we introduced the concept of a dynamic graph and our definition as an exponentially weighted average of the previous graph and a new edge set. We introduced a data structure that could be used to capture the evolution of a graph through time that was amenable to the exponential weighting scheme. This data structure allows the subgraph around any particular node to be quickly and efficiently expanded to an arbitrary diameter. Several applications were introduced that capitalized on this feature.

We have concentrated on the computational aspects of building and evolving the data structure for real applications. We have not explored the statistical aspects of treating our data structure and the associated algorithm for traversal as an approximation $\hat{\mathcal{G}}_t^k(d)$ to the true graph $\mathcal{G}_t$ where $k$ denotes the size of the top-$k$ edge set maintained in the data structure and $d$ the diameter employed by the traversal algorithm. We hope to initiate research to explore these ideas in the near future.

Another topic for further research is how to prune an extracted subgraph so that only informative edges and nodes are retained. A common approach from (static) graph theory is to extract the strongly connected component. However, we feel that certain features inherent to telecommunication networks such as asymmetric edges (due to some customers subscribing to a competitor), sinks (toll-free calling) and sources (large corporations), makes strongly connected components an inferior choice for pruning back a COI.

The updating and storage of the data structures are facilitated by the programming language Hancock, [Cortes *et al* (2000)]. Hancock is a domain-specific C-based language for efficient and reliable programming with transactional data. Hancock is publicly available for non-commercial use at

`http://www.research.att.com/~kfisher/hancock/`.

## References

**Kleinberg (1998)** Authoritative sources in a hyperlinked environment. *J. Kleinberg.* Proceedings 9th ACM-SIAM Symposium on Discrete Algorithms, 1998.

**Cortes and Pregibon (1999)** An information Mining Platform. *C. Cortes & D. Pregibon.* Proceedings of KDD99, San Diego, CA.

**Cortes *et al* (2000)** Hancock: A language for extracting signatures from data streams. *C. Cortes, K. Fisher, D. Pregibon, A. Rogers, & F. Smith.* Proceedings of KDD2000, Boston, MA.