Network Summarization

Aditya Bharadwaj
29 October 2015
Introduction

• Big Social Network

• Facebook Network – 1+Billion Nodes

• Is there a way to mine these large real networks efficiently?
Introduction

• Possible Solutions
  ▪ Sampling (Last lecture !!)
  ▪ Summarization
Introduction

• Summarization for large real networks
  - Find an equivalent smaller representation of the graph without losing any information.
  - Entire network should be known in advance
Overview

✓ Introduction

• **Fast Influence-based Coarsening for Large Networks**
  - Intuition
  - Problem Formulation
  - Efficient Algorithm
  - Evaluation

• **VoG: Summarizing and Understanding Large Graphs**
  - Common graph structures
  - Problem Formulation
  - VoG Algorithm
  - Experiments
Fast Influence-based Coarsening for Large Networks

Purohit et al
Intuition

• Most of the edges are not important.

• Power Law !!

• Merge such edges to obtain coarse representation.
Goal

Find an equivalent smaller representation of a large graph while preserving its diffusive properties.
Problem Formulation

• Independent Cascade Model
• \( G = (V, E, w) \): A directed, weighted graph
• Nodes can switch from inactive to active, but active nodes never inactivate
Problem Formulation

• **INPUT:** Weighted graph $G = (V, E, w)$ and a target fraction $0 < \alpha < 1$

• **GOAL:** Coarsen $G$ by repeatedly merging adjacent node pairs to obtain a weighted graph $H = (V', E', w')$ such that
  • $|V'| = (1 - \alpha) |V|$
  • Graph $H$ approximates $G$ with respect to its diffusive properties
Role of Eigenvalues

• RECAP: $\lambda$ was used to characterize the epidemic threshold
• If first eigenvalue of the coarsened graph $H$ is close to that of the original graph $G$, then $H$ approximates $G$ in terms of diffusive properties.
How to merge node pairs?
How to merge node pairs?
How to merge node pairs?
How to merge node pairs?

Diagram:

- Two paths are shown:
  - Path 1: 'x' -> 'e' -> 'd' -> 'b' -> 'a'
  - Path 2: 'x' -> 'e' -> 'd' -> 'c'

For Path 2, the edge 'd' to 'c' has a weight calculated as:

\[
\frac{0.5 	imes 0.5 + 0.5}{2}
\]
DEFINITION 4.2 (Merging node pairs). Let $Nb^i(v)$ (respectively $Nb^o(v)$) denote the set of in-neighbors (resp. out-neighbors) of a vertex $v$. Let $w(u, v)$ denote the weight of the corresponding edges. If the node pair $(a, b)$ is now contracted to a new vertex $c$, and $w(a, b) = \beta_1$ and $w(b, a) = \beta_2$, then the new edges are weighted as:

$$c_t^i = \begin{cases} 
\frac{(1 + \beta_2)a_t^i}{2} & \forall t \in Nb^o(a) \setminus Nb^o(b) \\
\frac{(1 + \beta_1)b_t^i}{2} & \forall t \in Nb^o(b) \setminus Nb^o(a) \\
\frac{(1 + \beta_2)(a_t^i) + (1 + \beta_1)(b_t^i)}{4} & \forall t \in Nb^o(a) \cap Nb^o(b)
\end{cases}$$
Merging Node Pairs
Merging Node Pairs

\[ c_i^x = \frac{a_i^x + a_i^x \beta_1}{2} \]
Merging Node Pairs
Graph Coarsening Problem

INPUT: Directed, strongly connected, weighted graph $G = (V, E, w)$ without self loops and a target fraction $0 < \alpha < 1$

OUTPUT: $E^* = \arg\min_{E' \subseteq E, |E'| = \alpha |V|} |\lambda_G - \lambda_{G'}|$, where $G'$ is obtained from $G$ by merging all node pairs in $E'$. 
Proposed Solution

• Score(a, b) = |\lambda_{G-(a,b)} - \lambda_G| = \Delta \lambda_{(a,b)}

• Greedy Heuristic Approach
Example

(a) Original Network
Example

(b) Assigning scores to each edge
Example

(c) Coarsened Network
Main Idea - COARSENET

• Can we approximate the scores of each node pair in linear time?

• Matrix Perturbation Argument to approximate the change in eigenvalue due to merging two adjacent nodes in constant time
Coarsening Algorithm - \textbf{COARSENET}

\textbf{Input:} A directed, weighted graph $G=(V,E,w)$, a reduction factor $\alpha$

\textbf{Output:} Coarsened graph $G_{coarse}^\alpha=(V',E',w')$

1: $i = 0$
2: $n = |V|$
3: $G' = G$
4: \textbf{for each} adjacent pair of nodes $a, b \in V$ \textbf{do}
5: \hspace{1em} Compute $score(a, b)$
6: \hspace{1em} $\pi \leftarrow$ ordering of node pairs in increasing order of $score$
7: \hspace{1em} \textbf{while} $i \leq \alpha n$ \textbf{do}
8: \hspace{2em} $(a, b) = \pi(i)$
9: \hspace{2em} $G' \leftarrow \text{Contract}_{G'}(a, b)$
10: \hspace{1em} $i++$
11: \textbf{return} $G_{coarse}^\alpha = G'$

\textit{Worst – case time complexity}

$O(mln(m) + \alpha nn_\theta)$
CSPIN: Influence Maximization Framework

• Three stages –

  • **Coarsen** the network graph $G$ to smaller graph $G_{coarsen}$

  • **Solve** Influence Maximization Problem to get $k$ vertices in the coarsened graph which optimizes the objective function.

  • **Pull back** the solutions on the original graph by selecting. Given a seed $s_i$ in $G_{coarse}$, select a vertex $v$ uniformly at random from $\mu^{-1}(s_i)$
Algorithm 2 CSPIN: Influence Maximization Framework

**Input:** A weighted graph $G=(V,E,w)$, the number of seeds $k$, a reduction factor $\alpha$

**Output:** A seed set $S$ of $k$ seeds

1. $G_{\text{coarse}}^\alpha, \mu \leftarrow \text{COARSEN\textsc{et}} (G, \alpha)$  \hspace{1cm} (See Algorithm 1)
2. $s'_1, s'_2, \ldots, s'_k \leftarrow \text{InfluenceMaximization}(G_{\text{coarse}}^\alpha, k)$
3. for $i = 1, \ldots, k$ do
4. \hspace{0.5cm} $s_i \leftarrow \text{random sample from } \mu^{-1}(s'_i)$
5. return $S = \{s_1, s_2, \ldots, s_k\}$
Evaluation

- Performance for the GCP problem
- Application 1: Influence Maximization
- Application 2: Diffusion Characterization
Performance for the GCP problem

(a) Amazon
(b) DBLP
(c) Brightkite

Effectiveness
Performance for the GCP problem

Scalability
Application 1: Influence Maximization

Effectiveness
Application 1: Influence Maximization

Scalability
Application 2: Diffusion Characterization

- Used a Flixster dataset to understand cascade datasets in exploratory setting.
- Distribution of movies over groups/supernodes
- Groups through the lens of surrogates
Overview

- Introduction
- Fast Influence-based Coarsening for Large Networks
  - Intuition
  - Problem Formulation
  - COARSENET Algorithm and CSPIN Framework
  - Evaluation
- VOG: Summarizing and Understanding Large Graphs
  - Common graph structures
  - Problem Formulation
  - VoG Algorithm
  - Experiments
VOG: Summarizing and Understanding Large Graphs
Koutra et al.
Common graph structures

- Construct a ‘vocabulary’ of such structures

Source: https://en.wikipedia.org
Example

- Original Wikipedia Controversy graph
- Nodes stand for Wikipedia contributors and edges link users who edited the same part of the article
Example

- Stars
- Hubs are editors or heavy contributors or admins
Example

- Bipartite Graphs
- Edit Wars!!
- Brazil vs Brasil
Problem Formulation

**Problem 1. (Informal)**

- **Given:** a graph
- **Find:** a set of possibly overlapping subgraphs
- *to most succinctly describe* the given graph, i.e., explain as many of its edges in as simple possible terms,
- *in a scalable* way, ideally linear on the number of edges.
How do you describe the given graph?

• Minimum Description Length (MDL)

• MDL = Model + Data
MDL for Graph Summarization

• What are our models?

• How our model describes data?

• How do we encode this description in bits?
Models

• Our model $M$ is ordered lists of graph structures

• Each structure $s \in M$ identifies a patch of Adjacency matrix $A$
area(s, M, A)
Models

- $C_x$ - set of all possible structures of type $x \in \Omega$
- $C = \bigcup_x C_x$
- Model Family $M$ consists of all possible permutations of all possible subsets of $C$. 
Graph Description

M = A (Unlikely!!)
Graph Description

MDL requires encoding to be lossless!!

Source: http://www.slideshare.net/SessionsEvents/danai-koutra
Graph Description

\[ L(G, M) = L(M) + L(E) \]

\[ E = M \oplus \hat{A}. \]
Minimum Graph Description Problem

Given a graph $G$ with adjacency matrix $A$, and the graph structure vocabulary $\Omega$, by the MDL principle we are after the smallest model $M$ for which the total encoded length

$$L(G, M) = L(M) + L(E)$$

is minimal, where $E = M \oplus A$ is the error matrix, and $M$ is an approximation of $A$ deduced by $M$. 
Encoding the Model - $L(M)$

\[ L(M) = L_N(|M| + 1) + \log \left( \frac{|M| + 1}{|\Omega| + 1} \right) + \sum_{s \in M} \left( -\log \Pr(x(s) \mid M) + L(s) \right) \]
Encoding for Full Cliques

\[ L(fc) = L_N(\|fc\|) + \log \left( \binom{n}{\|fc\|} \right) \]
Encoding for Near Cliques

\[ L(nc) = L_N(||nc||) + \log\left(\binom{n}{||nc||}\right) \]
\[ + \log(||\text{area}(nc)||) + ||nc||l_1 + ||nc||'l_0. \]

\[ l_1 = -\log\left(\frac{||nc||}{(||nc|| + ||nc||')}\right), \]

||nc|| - number of edges present in nc

||nc||' - number of edges present in nc
Encoding for Bipartite Cores

\[ L(fb) = L_N(|A|) + L_N(|B|) + \log \left( \frac{n}{|A|, |B|} \right), \]

\[ L(nb) = L_N(|A|) + L_N(|B|) + \log \left( \frac{n}{|A|, |B|} \right) + \log(|area(nb)|) + \|nb\|_{l_1} + \|nb\|'_{l_0}. \]
Encoding for Stars

\[ L(st) = L_{\mathbb{N}}(|st| - 1) + \log n + \log \left( \frac{n - 1}{|st| - 1} \right), \]
Encoding for Chains

\[ L(ch) = L_N(|ch| - 1) + \sum_{i=0}^{\vert ch \vert} \log(n - i), \]
Encoding the Error

Two types of error:

- $E^+$: Area of $A$ that $M$ does model, and for which $M$ includes superfluous edges.

- $E^-$: the area of $A$ not modeled by $M$, for which $M$ lacks edges
Encoding the Error

\[ L(E^+) = \log(|E^+|) + ||E^+||_1 + ||E^+||'_0 \]

\[ L(E^-) = \log(|E^-|) + ||E^-||_1 + ||E^-||'_0 \]
VoG Algorithm

Input: graph $G$

Step 1: Subgraph Generation. Generate candidate – possibly overlapping – subgraphs using one or more graph decomposition methods.

Step 2: Subgraph Labeling. Characterize each subgraph as a perfect structure $x \in \Omega$, or an approximate structure by using MDL to find the type $x$ that locally minimizes the encoding cost. Populate the candidate set $C$.

Step 3: Summary Assembly. Use the heuristics *Plain*, *Top10*, *Top100*, *Greedy’nForGet* (Sec. 4.3) to select a non-redundant subset from the candidate structures to instantiate the graph model $M$. Pick the model of the heuristic with the lowest description cost.

**return** graph summary $M$ and its encoding cost.
Subgraph Generation

Community Detection and Clustering Algorithms!!

Subgraph Labelling – Perfect Structures

- **Clique and Chain** – using the degree distribution

- **Bipartite Graph**
  - If magnitude of its maximum and minimum eigenvalues is same
  - To find the node ids in the two node sets, A and B, we use BFS (Breadth First Search) with node coloring.

- **Star** - If one of the node sets has size 1, then the given substructure is encoded as star.
Subgraph Labelling - Approximate Structures

• We encode the subgraph as each of the 6 candidate vocabulary structures, and choose the structure that has the lowest local encoding cost

• $m^*$ is a graph model with only one subgraph encoded as a structure.

$$L(m^*) + L(E_{m^*}^+) + L(E_{m^*}^-),$$
Labelling Approximate Structures

- **Clique**s - all nodes have the same structural role
- **Star** - Highest degree node is encoded as hub and rest of them as spokes
- **Bipartite core**
  - Max-cut problem
  - NP - Hard!!
  - Approximation algorithms - Fast Belief Propagation
- **Chain**
  - Longest Path Problem (NP Hard)
  - Employ heuristic
Summary Assembly

**Input:** Given candidate set $C$

**Goal:** Find $M \in \mathcal{M}$ such that $M$ minimizes the cost of encoding.

**Possible Solution:**
- Compute cost for all possible combinations. (Not scalable!!)
- We need heuristics.
Approximate Solution

**Encoding benefit**: #bits saved by encoding the subgraph by minimum cost structure instead of including them in error matrix

3 heuristics -
- PLAIN - Choose all candidates in C
- TOP-K - Choose top $k$ candidates in C
- GREEDY’NFORGET
GREEDY’NFORGET

• Include each candidate in C sequentially in $M$.

• If encoding cost doesn't increase, keep it in $M$; otherwise remove it.

• Works best for medium sized candidate set.
Experiments

• Are the real graphs structured, or random and noisy?

• What structures do the graph summaries consist of, and how can they be used for understanding?

• Is VOG scalable?
Are the real graphs structured, or random and noisy?

• ORIGINAL - The whole adjacency matrix is encoded as if it contains no structure; that is, \( M = \emptyset \), all of \( A \) is encoded through \( L(E^-) \)

• PLAIN
• TOP-10
• TOP-100
• GREEDY’NFORGET
Are the real graphs structured, or random and noisy?

<table>
<thead>
<tr>
<th>Graph</th>
<th>ORIGINAL (bits)</th>
<th>VoG PLAIN (u.e.:)</th>
<th>VoG TOP10 (u.e.:)</th>
<th>VoG TOP100 (u.e.:)</th>
<th>VoG GREEDY’NFORGET (u.e.:)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flickr</td>
<td>35 210 972</td>
<td>81% (4%)</td>
<td>99% (72%)</td>
<td>97% (39%)</td>
<td>95% (36%)</td>
</tr>
<tr>
<td>WWW-Barabasi</td>
<td>18 546 330</td>
<td>81% (3%)</td>
<td>98% (62%)</td>
<td>96% (51%)</td>
<td>85% (38%)</td>
</tr>
<tr>
<td>Epinions</td>
<td>5 775 964</td>
<td>82% (6%)</td>
<td>98% (65%)</td>
<td>95% (46%)</td>
<td>81% (14%)</td>
</tr>
<tr>
<td>Enron</td>
<td>4 292 729</td>
<td>75% (2%)</td>
<td>98% (77%)</td>
<td>93% (46%)</td>
<td>75% (6%)</td>
</tr>
<tr>
<td>AS-Oregon</td>
<td>475 912</td>
<td>72% (4%)</td>
<td>87% (59%)</td>
<td>79% (25%)</td>
<td>71% (12%)</td>
</tr>
<tr>
<td>Chocolate</td>
<td>60 310</td>
<td>96% (4%)</td>
<td>96% (70%)</td>
<td>93% (35%)</td>
<td>88% (27%)</td>
</tr>
<tr>
<td>Controversy</td>
<td>19 833</td>
<td>98% (5%)</td>
<td>94% (51%)</td>
<td>96% (12%)</td>
<td>87% (31%)</td>
</tr>
</tbody>
</table>
Are the real graphs structured, or random and noisy?

**Observation:**
Real graphs do have structure; VOG, with or w/o structure selection, achieves better compression than the ORIGINAL approach that assumes no structure.
What structures do the graph summaries consist of?

<table>
<thead>
<tr>
<th>Graph</th>
<th>PLAIN</th>
<th></th>
<th>TOP10</th>
<th></th>
<th>TOP100</th>
<th></th>
<th>GREEDY'NFORGET</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>st</td>
<td>nb</td>
<td>fc</td>
<td>fb</td>
<td>ch</td>
<td>nc</td>
<td>st</td>
</tr>
<tr>
<td>Flickr</td>
<td>24385</td>
<td>3750</td>
<td>281</td>
<td>9</td>
<td>-</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>WWW-Barabasi</td>
<td>10027</td>
<td>1684</td>
<td>487</td>
<td>120</td>
<td>26</td>
<td>-</td>
<td>9</td>
</tr>
<tr>
<td>Epinions</td>
<td>5204</td>
<td>528</td>
<td>13</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>9</td>
</tr>
<tr>
<td>Enron</td>
<td>3171</td>
<td>178</td>
<td>3</td>
<td>11</td>
<td>-</td>
<td>-</td>
<td>9</td>
</tr>
<tr>
<td>AS-Oregon</td>
<td>489</td>
<td>85</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>10</td>
</tr>
<tr>
<td>Chocolate</td>
<td>170</td>
<td>58</td>
<td>-</td>
<td>-</td>
<td>17</td>
<td>-</td>
<td>9</td>
</tr>
<tr>
<td>Controversy</td>
<td>73</td>
<td>21</td>
<td>-</td>
<td>1</td>
<td>22</td>
<td>-</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What structures do the graph summaries consist of?

Observation:
The summaries of all the selection heuristics consist mainly of stars, followed by near-bipartite cores. In some graphs, like Flickr and WWW-Barabasi, there is a significant number of full cliques.
Is VOG scalable?

The complexity of VOG is $O(m)$, i.e., near-linear on the number of edges of the input graph.