Diffusion Centrality

Chanhyun Kang
Motivation

• Social network influence analysis has received considerable attention

• Much research on how various diffusive properties (behaviors) spread through the networks

• Various diffusion models developed in various areas
  – Obviously, diffusion processes are different according to each diffusive property (behavior)

• Question!

Can we measure the influence of a vertex in a social network with respect to a given diffusion model?
Motivation

How to measure the influence of a vertex in a social network with respect to a given diffusion model?

• Centrality is widely used in social network analysis.
  – Ex. Degree, Eigenvector, Betweenness, Stress, Closeness and so on

• No centrality measures consider semantic aspects of networks
  – Does a man who doesn't have a baby have interest about a diaper complain article on Facebook and try to share it with friends? (vertex property: having a baby or not)
  – Do students recommend a latest fashion hat to their professors? (edge property: friend, supervisor, coworker and so on)

• No centrality measures consider diffusion processes explicitly
  – Information propagation and disease infection processes are different.
  – There are no centrality measures appropriate for infection and gossip processes ... Existing centrality measures are only valid for specific flow processes that they assume only *

* “Centrality and network flow” by Stephen P. Borgatti, Social Networks, 2005
Motivation

• We suggest a new notion of Diffusion Centrality to quantify the influence of vertices in a social network with respect to
  – A given diffusion model that explains how a diffusive property spreads in the network
  – The structural properties of the network
  – The semantic properties of the network
Motivation example

Organization O has historical data on the spread of some phenomenon \( p \) in a SN. They have identified a diffusion model for \( p \). They want to assign a score to each user \( u \) expressing how important user \( u \) is in diffusing the phenomenon \( p \).
Motivation example

The computed centrality measures of vertices

**Betweenness**

**Closeness**

**Eigenvector**

**Degree**
Motivation example

The expected number of infected vertices using top-3 vertices

Betweenness

Closeness

Eigenvector

Degree
Motivation example

When we used top-3 diffusion centrality vertices,
Quality of diffusion

<table>
<thead>
<tr>
<th>Measures</th>
<th>Top 1 person</th>
<th>Top 2 persons</th>
<th>Top 3 persons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diffusion Centrality</td>
<td>5.69</td>
<td>11.03</td>
<td>11.59</td>
</tr>
<tr>
<td>Eigenvector Centrality</td>
<td>5.34</td>
<td>6.34</td>
<td>7.34</td>
</tr>
<tr>
<td>Degree Centrality</td>
<td>1.964</td>
<td>3.264</td>
<td>7.99</td>
</tr>
<tr>
<td>Closeness Centrality</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Betweenness centrality</td>
<td>1</td>
<td>2.5</td>
<td>3.8</td>
</tr>
</tbody>
</table>
Diffusion Model: Syntax

• Set of condition probability statements such as:

  i. \[ P(\text{REC}(X) \mid \text{friend}(Y, X, W) \land W=1 \land \text{REC}(Y)>0 \land \text{SEX}(Y)=\text{SEX}(X) \land \text{LIKE}\_\text{RUNNING}(Y) \land \text{LIKE}\_\text{RUNNING}(X)) = 0.8 \times \text{REC}(Y). \]

  If Y is a friend of X with weight 1, the probability that Y recommends the product exceeds 0, X and Y are of the same sex and both X,Y like running, then the probability that X will recommend the product is 80% that of Y.

  REC(X) : recommendation probability for a product, a diffusive property

  ii. \[ P(\text{REC}(X) \mid \text{friend}(Y, X, W) \land W=1 \land \text{REC}(Y)>0 \land \text{SEX}(Y) \neq \text{SEX}(X) \land \text{LIKE}\_\text{RUNNING}(Y) \land \text{LIKE}\_\text{RUNNING}(X)) = 0.5 \times \text{REC}(Y) \]

  iii. \[ P(\text{REC}(X) \mid \text{friend}(Y, X, W) \land W=1 \land \text{REC}(Y)>0 \land \text{SEX}(Y)=\text{SEX}(X) \land \text{LIKE}\_\text{RUNNING}(X)) = 0.2 \times \text{REC}(Y) \]

  Diffusion models can be readily learned from historical data using standard learning algorithms.
Diffusion Fixpoint Operator

- $\ell$ is a “labeling” assigning a probability that a vertex $v$ has diffusive property $p$.
- $T_{S,D}(\ell)(v)$ is the probability that vertex $v$ has diffusive property $p$ after applying all rules in diffusion model $D$ once.

$$T_{S,D}(\ell)(v) = \max(\{\ell(v)\} \cup \{c \times \prod_{p(v') \in \text{body}(r)} \ell(v') \mid \exists r \in \text{grd}(D) \text{ s.t. } r \text{ is enable and of the form } P(p(v) | A_1 \land \cdots \land A_n) = c\})$$

After applying all rules of $D$ for $v$ once,

$$p(v_1) = \max\{p(v_1), p(v_1)xc_1, p(v_2)xc_2, p(v_3)x p(v_4)xc_3\}$$

We assume the probabilities are independent!
Diffusion Fixpoint Operator

• If we apply $T_{S,D}$ for all vertices once, we can get the probabilities that vertices have diffusive property $p$ after first contagion process.

• We keep applying $T_{S,D}$ for all vertices continuously until no changes of probabilities occur.

• Sum of probabilities at the stable state is the expected number of infected vertices in social network $S$ w.r.t. diffusion model $D$.

Can we ensure that we can reach the stable state?
**Diffusion Fixpoint Operator**

- $\ell$ is a "labeling" assigning a probability that a vertex $v$ has diffusive property $p$.
- $T_{S,D}(\ell)(v)$ is the probability that vertex $v$ has diffusive property $p$ after applying all rules in diffusion model $D$ once.

\[
T_{S,D}(\ell)(v) = \max\{\ell(v)\} \cup \{c \times \prod_{p(v') \in \text{body}(r)} \ell(v') \mid \exists r \in \text{grd}(D) \text{ s.t. } r \text{ is enable and of the form } P(p(v)|A_1 \land \cdots \land A_n) = c\}
\]

- **Theorem:** $T_{S,D}$ is a monotonic operator that has a least fixpoint, $\text{lfp}(T_{S,D})$
- **Theorem:** $\text{lfp}(T_{S,D})$ can be obtained by
  - Starting with the labeling that assigns 0 to all vertices (except for those originally labeled as having diffusive property $p$).
  - Applying the $T_{S,D}$ operator till no change occurs.
Section Outline

• Motivation: Classical centrality measures fall short
• Diffusion Model
  – Syntax
  – Fixpoint Operator
• **Diffusion Centrality (DC) Definition**
• Hypergraph-based DC Computation Algorithm
• Experimental Results
• Conclusion
Diffusion Centrality Definition

- Diffusion Centrality $dc(v)$ of vertex $v$ is obtained as follows:
  - Compute expected number of infected vertices if $v$ has diffusive property $p$;
  - Compute expected number of infected vertices if $v$ does NOT have diffusive property $p$;
  - Take the difference.

$$dc(v) = \sum_{v' \in V - \{v\}} lfp(T_{S \oplus p, D})(v') - \sum_{v'' \in V - \{v\}} lfp(T_{S \ominus p, D})(v'')$$
DC Computation: Key Insight, I

- Let $F = \text{lfp}(T_{S,D})$. If we pre-compute $F$, then to compute $dc(v)$, we only need to compute one of the two fixpoints.
  - If vertex $v$ originally had diffusive property $p$, then:
    \[
    dc(v) = \sum_{v' \in V - \{v\}} F(v') - \sum_{v' \in V - \{v\}} \text{lfp}(T_{S \oplus p, D})(v'')
    \]
  - If vertex $v$ originally didn’t have diffusive property $p$:
    \[
    dc(v) = \sum_{v' \in V - \{v\}} \text{lfp}(T_{S \oplus p, D})(v') - \sum_{v' \in V - \{v\}} F(T_{S \oplus p, D})(v'')
    \]

# of lfp computations to compute diffusion centrality of all vertices is $n+1$ (not $2n$), where $n = |V|$
DC Computation: Key Insight, II

- We can eliminate a bunch of rules.
- All instantiated diffusion rules have $p(-)$ in the head.
- If $q(x)$ occurs in the rule body and $q \neq p$ and vertex $x$ does not have property $q$, then get rid of the rule.
- If $e(x,y,w)$ occurs in the rule body and edge$(x,y)$ does not have the desired weight, then get rid of the rule.
- Otherwise, get rid of all non-$p$ atoms in the rule body.

If rule instances (except diffusive property $p$ in body) are not satisfiable in $S$, the rules will never be satisfied in $lfp$ computation. Let’s remove the rule instances.
DC Computation: Key Insight, III

• From the remaining rule instances, we can create a hypergraph $H$
  – Hyperedge $h: (S, t)$ draws an arc from a set of vertices to a vertex. The weight is the probability of the rule instance.
  – Source $S$: All vertices of the form $p()$ in the rule body.
  – Target $t$: The rule head.

• $lfp$ computation can be computed by applying hyperedges in descending order of edge weights.
  – Larger probabilities are assigned to vertices first.
  – Our $lfp$ computation ensures that the assigning is once for each vertex.

• Our HyperDC() algorithm finds all vertices with DC exceeding a threshold $\tau$. We incorporate a set of methods to prune based on $\tau$.

• Complexity of HyperDC() : $O(|N| + |H| \cdot (\log |H| + u_{\text{max}} \cdot S_{\text{max}}))$
  
  where $u_{\text{max}} = \max_{v \in V}||\{ h \mid h \in H \land v \in S(h) \}||$ and $S_{\text{max}} = \max_{h \in H}||S(h)||$
Experiments

• YouTube data with diffusion model for joining specific YouTube groups. (Users originally have a group and have ‘friend’ relationships)

• Diffusion model: \( P(g(Y) \mid \text{friend}(Y,X) \& g(X)>0 \& q(X)) = \rho \times g(X) \)
  – If \( X \) is a member of \( g \), \( X \) is a friend of \( Y \) and \( X \) has property \( q \), then \( Y \) is a member of \( g \) with probability \( \rho \). (*Flickr model*)
  – We randomly selected a group \( g \) as a diffusive property for each run

• Varied size of networks in YouTube data
  – From 20K to 100K vertices in steps of 20K.
  – From 100K to 1M vertices in steps of 100K.

• Ran on Intel Xeon@2.40 GHz, 24 GB RAM

Only a snapshot of our experimental results are listed here.

*“A measurement-driven analysis of information propagation in the flickr social network”, Meeyoung Cha et al, 2009*
Running Time (in milliseconds)

- The average time for one vertex when 2.5%* of vertices have property $q$
  (In Flickr influence analysis research*, vertices infected 1-2% of their neighbors)

<table>
<thead>
<tr>
<th>Number of vertices in the SN</th>
<th>20K</th>
<th>40K</th>
<th>60K</th>
<th>80K</th>
<th>100K</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Degree</strong></td>
<td>0.03</td>
<td>0.07</td>
<td>0.11</td>
<td>0.14</td>
<td>0.31</td>
</tr>
<tr>
<td><strong>Eigenvector</strong></td>
<td>0.42</td>
<td>0.43</td>
<td>0.60</td>
<td>0.68</td>
<td>0.91</td>
</tr>
<tr>
<td><strong>Diffusion</strong></td>
<td>8.71</td>
<td>17.77</td>
<td>29.36</td>
<td>44.20</td>
<td>53.08</td>
</tr>
<tr>
<td><strong>Betweenness</strong></td>
<td>88.80</td>
<td>231.57</td>
<td>424.52</td>
<td>581.56</td>
<td>781.60</td>
</tr>
</tbody>
</table>

Unsurprisingly, degree and eigenvector centrality can be computed very fast. But diffusion centrality is at least 1 order of magnitude times faster than betweenness centrality.

When even 20% of vertices have property $q$, the average time for a vertex is less than 0.3 sec with 100K vertices and it is faster than betweenness (0.78 sec with 100K vertices)

*“A measurement-driven analysis of information propagation in the flickr social network”, Meeyoung Cha et al, 2009
Running Time (in milliseconds)

- The average time for one vertex when 2.5%* of vertices have property \( q \)
  (In Flickr influence analysis research*, a vertex infected 1-2% of its neighbors)

*“A measurement-driven analysis of information propagation in the flickr social network”, Meeyoung Cha et al, 2009
Quality of Results

- We assigned a diffusive property (a group) to top-k vertices w.r.t. each centrality

\[ Y\text{-axis} = \frac{\text{the spread after assigning diffusive properties}}{\text{the spread in the original social network}} \]

(Remember that original networks also have diffusive properties)

When we assigned diffusion properties using diffusion centrality, the expected number of infected vertices is 30-550 times larger than the expected number in original network.
Questions