Evolving Centralities in Temporal Graphs: a Twitter Network Analysis

Fabíola S. F. Pereira¹, Sandra de Amo¹ and João Gama²

¹Federal University of Uberlândia, Brazil
²INESC TEC, University of Porto, Portugal

MobDM – Jun 2016
Temporal Graphs

A representation that encodes **temporal data** into graphs
Temporal Graphs

Disease Spreading

If A has a **virus**, which nodes were infected?
Temporal Graphs

Rumor Spreading

If A creates a rumor, which people will hear about it?
Temporal Graphs vs Static Graphs
Outline

✓ **Twitter as a Temporal Network**

✓ **Temporal Paths**

✓ **Centrality Metrics in Temporal Networks**

✓ **Experiments on Twitter Dataset**
Twitter as a Temporal Network

![Diagram of a temporal graph with nodes A, B, C, and D, and edges with time intervals like [1,9], [2,4], [1,4], [6,8], [4,7].]
Twitter as a Temporal Network

\[ [\text{init}, \text{end}] \]

B follows A (information flow) from \( t_{\text{init}} \) to \( t_{\text{end}} \)
Twitter as a Temporal Network

Following a user at any time:

- $A_{[t_{init}, t_{end}]} \rightarrow B$

B follows A (information flow) from $t_{init}$ to $t_{end}$

- $W = [n, N] = \text{observation window}$
Twitter as a Temporal Network

B follows A (information flow) from $t_{\text{init}}$ to $t_{\text{end}}$

$W = [n,N] = \text{observation window}$

$R = \text{retention time} = 1 \text{ day}$
Twitter as a Temporal Network

B follows A (information flow) from $t_{\text{init}}$ to $t_{\text{end}}$

- $W = [n,N]$ = observation window
- $R = \text{retention time} = 1 \text{ day}$
- $T = \text{edge traversal time} = 0$
Temporal Metrics
Temporal Metrics

The concept of geodesic distance cannot be limited to the number of hops separating two nodes but should also take into account the temporal ordering of links.
Temporal Path

<table>
<thead>
<tr>
<th>Temporal Path</th>
<th>Duration</th>
<th>Is fastest path?</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(B,E) = ((B,C,1), (C,E,2))</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>P(B,E) = ((B,D,7), (D,E,9))</td>
<td>2</td>
<td>no</td>
</tr>
<tr>
<td>P(B,E) = ((B,D,7), (D,E,8))</td>
<td>1</td>
<td>yes</td>
</tr>
</tbody>
</table>

W = [1,20]
R = 1 day
T = 0
Temporal Metrics

Revisiting centrality metrics...

\[
closeness(v) = \sum_{u \in V \setminus \{v\}} \frac{1}{\min(d_{P,v,u})}
\]

✓ How close a node is from the others nodes in the graph
✓ High closeness = best visibility into what is happening
Temporal Metrics

Revisiting centrality metrics....

\[
closeness(v) = \sum_{u \in V \setminus \{v\}} \frac{1}{\min(d_{P,v,u})}
\]

- How close a node is from the others nodes in the graph
- High closeness = best visibility into what is happening
Temporal Metrics

Revisiting centrality metrics...

\[
\text{betweenness}(v) = \sum_{v \neq j \neq k} \frac{w_v(j, k)}{w(j, k)}
\]

- High betweenness = great influence over what flows
- High betweenness = control the flow of information (gatekeeper)
Temporal Metrics

Revisiting centrality metrics...

\[ \text{betweenness}(v) = \sum_{v \neq j \neq k} \frac{w_v(j, k)}{w(j, k)} \]

- High betweenness = great influence over what flows
- High betweenness = control the flow of information (gatekeeper)
Betweenness centrality

<table>
<thead>
<tr>
<th>Node</th>
<th>Fastest Path (temporal)</th>
<th>Shortest Path (static)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1.33</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>0.66</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

W = [1,20]
R = 1 day
T = 0
Betweenness centrality

Fastest Path (temporal) | Nodes Ranking | Rank
---|---|---
3 | E | 1
1.33 | C | 2
0.66 | D | 3
0 | B | 4

Shortest Path (static)

Nodes Ranking | Shortest Path (static)
---|---
1 | B | 4
1 | E | 4
3 | C | 2
3 | D | 2

W = [1,20]
R = 1 day
T = 0
Betweenness centrality

R = 1 day
T = 0

<table>
<thead>
<tr>
<th>Ranking</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>E, C</td>
</tr>
<tr>
<td>3</td>
<td>A, B, D, F</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Betweenness centrality

**Diagram**

- **Nodes:** A, B, C, D, E, F
- **Edges:**
  - A to C: [1,2]
  - B to C: [1,2]
  - C to D: [7,9]
  - C to E: [1,2]
  - D to E: [8,9]
  - E to F: [2,4], [11,14]

**Table**

<table>
<thead>
<tr>
<th>Ranking</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>E, C</td>
</tr>
<tr>
<td>3</td>
<td>A, B, D, F</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Tables**

<table>
<thead>
<tr>
<th>Ranking</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>E</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>4</td>
<td>A, B, F</td>
</tr>
</tbody>
</table>

**Time intervals**

- **W:**
  - [1,7] at t = 7
  - [1,14] at t = 14
  - [1,21] at t = 21

**Parameters**

- R = 1 day
- T = 0
Betweenness centrality

Ranking | Node
---|---
1 | E, C
3 | A, B, D, F
- | -
- | -

Ranking | Node
---|---
1 | E
2 | C
3 | D
4 | A, B, F

Ranking | Node
---|---
1 | E
2 | C
3 | D
4 | A, B, F

R = 1 day
T = 0

W = [1,7]
W = [1,14]
W = [1,21]
Network Evolution
Computation
How to compute temporal metrics?

Stream representation of a temporal network
How to compute temporal metrics?

Stream representation of a temporal network

The data stream representation of an interval graph is a sequence of contacts from all edges in $G$, ordered by the time of each possible contact.
How to compute temporal metrics?

Stream representation of a temporal network

\{(v1, v2, 4, 6), (v1, v3, 1, 3),
(v2, v4, 5, 6), (v4, v5, 6, 8)\}
How to compute temporal metrics?

Stream representation of a temporal network

\[\{(v_1, v_2, 4, 6), (v_1, v_3, 1, 3), (v_2, v_4, 5, 6), (v_4, v_5, 6, 8)\} \rightarrow \{(v_1, v_3, 1), (v_1, v_3, 2), (v_1, v_3, 3), \ldots\}\]
How to compute temporal metrics?

Stream representation of a temporal network

\{(v_1, v_2, 4, 6), (v_1, v_3, 1, 3), (v_2, v_4, 5, 6), (v_4, v_5, 6, 8)\} → \{(v_1, v_3, 1), (v_1, v_3, 2), (v_1, v_3, 3), (v_1, v_2, 4), \ldots\}
How to compute temporal metrics?

Stream representation of a temporal network

\{(v1, v2, 4, 6), (v1, v3, 1, 3), (v2, v4, 5, 6), (v4, v5, 6, 8)\} \rightarrow \{(v1, v3, 1), (v1, v3, 2), (v1, v3, 3), (v1, v2, 4), (v2, v4, 5), (v1, v2, 5), (v1, v2, 6), (v2, v4, 6), (v4, v5, 6), (v4, v5, 7), (v4, v5, 8)\}
How to compute temporal metrics?

Temporal graph $G = (V,E)$ in its stream representation

$W, R, T$

All pairs fastest paths detection algorithm

All pairs fastest paths
How to compute temporal metrics?

Temporal graph \( G = (V,E) \) in its stream representation

\[ W, R, T \]

All pairs fastest paths detection algorithm

All pairs fastest paths

Betweenness Values

Centrality computation

Closeness Values
Twitter Dataset
Dataset

We need:

- Temporal dataset
- Dataset containing *when* edges start and end

Available temporal datasets:

- [Tang et al. 2010] Enrol Mail Dataset: an email dataset
- [Viswanath et al. 2009] Facebook Friendships and Posts:
  - Does not have *when* the links end (*just when they started*)
- [Cattuto et al. 2013] Contact network
Once stored entire observation network in file format, the Update Crawler is started. Its function is to update structural node information based on a given time interval $U$. For example, for $U = 24$ hs, the temporal network is built with one day granularity. Since structural information is not available in Twitter (Twitter API does not provide historical information about when a user starts/end following other), our dataset only makes sense from the moment Update Crawler is started. So, the observation network built from Data Crawler is our initial state. Table 3 details statistics of our dataset.

<table>
<thead>
<tr>
<th>Observation window $W$</th>
<th>[08/28/2015, 12/15/2015]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update window (granularity) $U$</td>
<td>1 day</td>
</tr>
<tr>
<td>Max fanout ($m$)</td>
<td>10000</td>
</tr>
<tr>
<td>MAX_LEVEL</td>
<td>1</td>
</tr>
<tr>
<td># nodes</td>
<td>144975</td>
</tr>
<tr>
<td># edges in first day</td>
<td>837961</td>
</tr>
<tr>
<td># total temporal edges</td>
<td>1222118</td>
</tr>
<tr>
<td>Avg # new follows/day</td>
<td>3492</td>
</tr>
<tr>
<td>Avg # unfollows/day</td>
<td>3657</td>
</tr>
<tr>
<td># seeds ($s$)</td>
<td>27</td>
</tr>
<tr>
<td># themes</td>
<td>9 (3 seeds each)</td>
</tr>
<tr>
<td>Themes related to seeds</td>
<td>politics, sport, news, religion, music, humor, fashion, health, TV</td>
</tr>
</tbody>
</table>
New follows/unfollows per week
Number of active edges
Twitter Network Evolution
Varying Time Intervals

<table>
<thead>
<tr>
<th>Period</th>
<th># of intervals</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weekly (WE)</td>
<td>15</td>
<td>(WE_1 = [09/01, 09/07], WE_2 = [09/08, 09/14], \ldots, WE_{15} = [12/08, 12/14])</td>
</tr>
<tr>
<td>Fortnightly (FO)</td>
<td>7</td>
<td>(FO_1 = [09/01, 09/15], FO_2 = [09/16, 09/30], \ldots, FO_7 = [11/30, 12/14])</td>
</tr>
<tr>
<td>Monthly (MO)</td>
<td>3</td>
<td>(MO_1 = [09/01, 09/30], MO_2 = [10/01, 10/31], MO_3 = [11/01, 11/30])</td>
</tr>
<tr>
<td>Total (TO)</td>
<td>1</td>
<td>(TO = [08/28, 12/15])</td>
</tr>
</tbody>
</table>

TABLE 3. Observation Windows

We measure the execution time of Algorithm 1 for different observation windows. Figure 4(a) illustrates these results. The values for \(WE, FO\) and \(MO\) correspond to the average execution time in each group of windows. We can observe an exponential behavior compatible with the increasing number of incoming contacts \(C\) (Figure 4(b)).

Our algorithm is dependent on the size of the observation window. In Figure 4(c) we show how different the fastest paths values can be just varying observation windows. This endorses the time-varying aspect of Twitter network.

Figure 4. Results varying the size of observation window when running all pairs fastest paths algorithm

5.2. Evolving Centralities

The closeness centrality can be easily calculated from the return of Algorithm 1. With all pairs fastest paths and their respective duration, the closeness \((v)\) is a straight sum of these values (see Eq. 1). In Figure 5(a) we can see the closeness value averaged across all users for different observation windows. The observation during short or large amount of time does not influence on closeness. These values depend mainly of the network behavior: as Twitter network is diversified and extremely dynamic, nodes’ closeness vary accordingly.

Another interesting analysis is illustrated in Figure 5(b). Three users \(u_1, u_2, u_3\) were randomly selected and their closeness analyzed over fortnightly intervals. Remark that these users are not seeds. The graph shows that users are always changing their closeness.

As well as closeness, the betweenness centrality is a straight calculus from the fastest paths returned by Algorithm 1 (see Eq. 2). Furthermore, the nodes have the same behavior in varying their centrality values. In Figure 5(a) we have the betweenness averaged across all users for different intervals. And in 5(c) the variation for users \(u_1, u_2, u_3\).

Finally, we rank all nodes according to their centrality values (first positions for higher centralities). For this analysis we consider a sequence of incremental observation windows of the type \(I_1 = [\text{day} 1, \text{day} 15], I_2 = [\text{day} 16, \text{day} 30], \ldots, I_7 = [\text{day} 91, \text{day} 105]\). The values in Table 4 suggest that nodes centralities are fairly dynamic and from one observation to the next, the node may have become more or less important.

Figure 5. Evolving centralities observations

6. Discussion

We consider the strategy proposed in this paper for mining all pairs fastest paths as a pseudo-stream mining. The Algorithm 1 is extremely sensitive to the size of the stream, despite processing data as stream. Its high velocity...
Avg Execution Time to compute fastest paths

Period | # of intervals | Values
---|---|---
Weekly (WE) | 15 | WE 1 = [09/01, 09/07], WE 2 = [09/08, 09/14], ..., WE 15 = [12/08, 12/14]

Fortnightly (FO) | 7 | FO 1 = [09/01, 09/15], FO 2 = [09/16, 09/30], ..., FO 7 = [11/30, 12/14]

Monthly (MO) | 3 | MO 1 = [09/01, 09/30], MO 2 = [10/01, 10/31], MO 3 = [11/01, 11/30]

Total (TO) | 1 | TO = [08/28, 12/15]

Table 3. Observation Windows

We measure the execution time of Algorithm 1 for different observation windows. Figure 4(a) illustrates these results. The values for WE, FO and MO correspond to the average execution time in each group of windows. We can observe an exponential behavior compatible with the increasing number of incoming contacts (Figure 4(b)).

Our algorithm is dependent on the size of the observation window. In Figure 4(c) we show how different the fastest paths values can be just varying observation windows. This endorses the time-varying aspect of Twitter network.

5.2. Evolving Centralities

The closeness centrality can be easily calculated from the return of Algorithm 1. With all pairs fastest paths and their respective duration, the closeness \((v)\) is a straight sum (see Eq. 1). In Figure 5(a) we can see the closeness value averaged across all users for different observation windows. The observation during short or large amount of time does not influence on closeness. These values depend mainly of the network behavior: as Twitter network is diversified and extremely dynamic, nodes' closeness vary accordingly.

Another interesting analysis is illustrated in Figure 5(b). Three users \(u_1, u_2\) and \(u_3\) were randomly selected and their closeness analyzed over fortnightly intervals. Remark that these users are not seeds. The graph shows that users are always changing their closeness.

As well as closeness, the betweenness centrality betweenness \((v)\) is a straight calculus from the fastest paths returned by Algorithm 1 (see Eq. 2). Furthermore, the nodes have the same behavior in varying their centrality values. In Figure 5(a) we have the betweenness averaged across all users for different intervals. And in 5(c) the variation for users \(u_1, u_2\) and \(u_3\).

Finally, we rank all nodes according to their centralitiy values (first positions for higher centralities). For this analysis we consider a sequence of incremental observation windows of the type \(I_1 = [day_1, day_15], I_2 = [day_16, day_30], ..., I_7 = [day_91, day_105]\). The values in Table 4 suggest that nodes centralities are fairly dynamic and from one observation to the next, the node may have become more or less important.
Avg Fastest Paths Duration

<table>
<thead>
<tr>
<th>Period</th>
<th># of intervals</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE</td>
<td>15</td>
<td>WE1 [09/01, 09/07], WE2 [09/08, 09/14], ..., WE15 [12/08, 12/14]</td>
</tr>
<tr>
<td>FO</td>
<td>7</td>
<td>FO1 [09/01, 09/15], FO2 [09/16, 09/30], ..., FO7 [11/30, 12/14]</td>
</tr>
<tr>
<td>MO</td>
<td>3</td>
<td>MO1 [09/01, 09/30], MO2 [10/01, 10/31], MO3 [11/01, 11/30]</td>
</tr>
<tr>
<td>TO</td>
<td>1</td>
<td>TO [08/28, 12/15]</td>
</tr>
</tbody>
</table>

TABLE 3. OBSERVATION WINDOWS

We measure the execution time of Algorithm 1 for different observation windows. Figure 4(a) illustrates these results. The values for WE, FO, and MO correspond to the average execution time in each group of windows. We can observe an exponential behavior compatible with the increasing number of incoming contacts C (Figure 4(b)).

Our algorithm is dependent on the size of the observation window. In Figure 4(c) we show how different the fastest paths values can be just varying observation windows. This endorses the time-varying aspect of Twitter network.

Figure 4. Results varying the size of observation window when running all pairs fastest paths algorithm

5.2. Evolving Centralities

The closeness centrality can be easily calculated from the return of Algorithm 1. With all pairs fastest paths and their respective duration, the closeness \( c(v) \) is a straight sum of these values (see Eq. 1). In Figure 5(a) we can see the closeness value averaged across all users for different observation windows. The observation during short or large amount of time does not influence on closeness. These values depend mainly of the network behavior: as Twitter network is diversified and extremely dynamic, nodes' closeness vary accordingly.

Another interesting analysis is illustrated in Figure 5(b). Three users \( u_1, u_2, \) and \( u_3 \) were randomly selected and their closeness analyzed over fortnightly intervals. Remark that these users are not seeds. The graph shows that users are always changing their closeness.

As well as closeness, the betweenness centrality

\[ \text{betweenness}(v) \]

is a straight calculus from the fastest paths returned by Algorithm 1 (see Eq. 2). Furthermore, the nodes have the same behavior in varying their centrality values. In Figure 5(a) we have the betweenness averaged across all users for different intervals. And in 5(c) the variation for users \( u_1, u_2, \) and \( u_3 \).

Finally, we rank all nodes according to their centrality values (first positions for higher centralities). For this analysis we consider a sequence of incremental observation windows of the type

\[ I_1 = [\text{day} 1, \text{day} 15], I_2 = [\text{day} 16, \text{day} 30], \ldots, I_7 = [\text{day} 91, \text{day} 105]. \]

The values in Table 4 suggest that nodes centralities are fairly dynamic and from one observation to the next, the node may have become more or less important.

Figure 5. Evolving centralities observations

6. Discussion

We consider the strategy proposed in this paper for mining all pairs fastest paths as a pseudo-stream mining. The Algorithm 1 is extremely sensitive to the size of the stream, despite processing data as stream. Its high velocity
Evolving Centralities

We define different intervals to perform the analysis. As Twitter dataset has been collected from Aug 28th, 2015 to Dec 15th, 2015, in table 3 we summarize the adopted observation windows.

TABLE 3. OBSERVATION WINDOWS

<table>
<thead>
<tr>
<th>Period</th>
<th># of intervals</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weekly (WE)</td>
<td>15</td>
<td>WE1 = [09/01, 09/07], WE2 = [09/08, 09/14], ..., WE15 = [12/08, 12/14]</td>
</tr>
<tr>
<td>Fortnightly (FO)</td>
<td>7</td>
<td>FO1 = [09/01, 09/15], FO2 = [09/16, 09/30], ..., FO7 = [11/30, 12/14]</td>
</tr>
<tr>
<td>Monthly (MO)</td>
<td>3</td>
<td>MO1 = [09/01, 09/30], MO2 = [10/01, 10/31], MO3 = [11/01, 11/30]</td>
</tr>
<tr>
<td>Total (TO)</td>
<td>1</td>
<td>TO = [08/28, 12/15]</td>
</tr>
</tbody>
</table>

We measure the execution time of Algorithm 1 for different observation windows. Figure 4(a) illustrates these results. The values for WE, FO and MO correspond to the average execution time in each group of windows. We can observe an exponential behavior compatible with the increasing number of incoming contacts (Figure 4(b)). Our algorithm is dependent on the size of the observation window.

In Figure 4(c) we show how different the fastest paths values can be just varying observation windows. This endorses the time-varying aspect of Twitter network.

5.2. Evolving Centralities

The closeness centrality can be easily calculated from the return of Algorithm 1. With all pairs fastest paths and their respective duration, the closeness \( (v) \) is a straight sum of these values (see Eq. 1). In Figure 5(a) we can see the closeness value averaged across all users for different observation windows. The observation during short or large amount of time does not influence on closeness. These values depend mainly of the network behavior: as Twitter network is diversified and extremely dynamic, nodes' closeness vary accordingly.

Another interesting analysis is illustrated in Figure 5(b). Three users \( u_1, u_2, u_3 \) were randomly selected and their closeness analyzed over fortnightly intervals. Remark that these users are not seeds. The graph shows that users are always changing their closeness.

As well as closeness, the betweenness centrality \( (v) \) is a straight calculus from the fastest paths returned by Algorithm 1 (see Eq. 2). Furthermore, the nodes have the same behavior in varying their centrality values. In Figure 5(a) we have the betweenness averaged across all users for different intervals. And in 5(c) the variation for users \( u_1, u_2, u_3 \).

Finally, we rank all nodes according to their centralities (first positions for higher centralities). For this analysis we consider a sequence of incremental observation windows of the type \( I_1 = [day_1, day_15], I_2 = [day_16, day_30], ..., I_7 = [day_91, day_105] \). The values in Table 4 suggest that nodes centralities are fairly dynamic and from one observation to the next, the node may have become more or less important.

Figure 5. Evolving centralities observations

6. Discussion

We consider the strategy proposed in this paper for mining all pairs fastest paths as a pseudo-stream mining. The Algorithm 1 is extremely sensitive to the size of the stream, despite processing data as stream. Its high velocity...
Evolving Centralities

Figure 5. Evolving centralities observations
Analyzing evolving centralities in networks

Our findings have shown that analyzing Twitter as a temporal network is different from just considering static analysis. We have developed an algorithm to perform analysis over Twitter follower/followee network. For mining streams of temporal networks and have used it shown how to compute closeness and betweenness centralities using fastest paths. We have developed an algorithm shown how to compute closeness and betweenness centralities of shortest path considering the time dimension. We have modeled evolving network structure viewpoint. We have modeled.

7. Conclusion

Applications

Our purpose in this paper was to analyze Twitter from knowledge, there is not a solution for exactly centrality and sampling methods [5], [7], [9], [15]. To best of our look in temporal networks over a real dataset from evolving problem [14]. On the other hand, our proposal is a first processing depends on the size of available memory, which can be applied in diverse real applications. We highlight here can be applied in this problem and tracking evolving centralities exactly people getting infected, but who infected them [16].

influence and disease) where the interest is in observing not two of them. First, the problem of contagion (information, can be applied in this problem as an application. For instance, these patterns help understand how users' preferences evolve over time for more accurate recommendation systems. The analysis of evolving centralities can reveal patterns of influence and communications in social networks. For instance, these patterns help understand how users' preferences evolve over time for more accurate recommendation systems.

Figure 1. Ranking positions variation for cumulative observations in window size $I$.

<table>
<thead>
<tr>
<th>User</th>
<th>$I_1$</th>
<th>$I_2$</th>
<th>$I_3$</th>
<th>$I_4$</th>
<th>$I_5$</th>
<th>$I_6$</th>
<th>$I_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_1$</td>
<td>544</td>
<td>530</td>
<td>533</td>
<td>530</td>
<td>544</td>
<td>580</td>
<td>600</td>
</tr>
<tr>
<td>$U_2$</td>
<td>653</td>
<td>644</td>
<td>600</td>
<td>589</td>
<td>592</td>
<td>615</td>
<td>617</td>
</tr>
<tr>
<td>$U_3$</td>
<td>122</td>
<td>123</td>
<td>100</td>
<td>224</td>
<td>220</td>
<td>235</td>
<td>249</td>
</tr>
</tbody>
</table>
Conclusions
Conclusions

✓ Twitter as a temporal network

✓ Revisit centrality metrics with temporal concepts

✓ Calculate evolving centralities in temporal graphs streams

✓ Perform experiments on Twitter follower/followee dataset
Future directions

- Comparison with static networks counterpart
- Analysis of user preferences and behaviors over time
- Analysis of the problem of contagion (information, influence)
- ...


Evolving Centralities in Temporal Graphs: a Twitter Network Analysis

Fabíola S. F. Pereira, Sandra de Amo and João Gama

www.lsi.facom.ufu.br/~fabiola
fabiola.pereira@ufu.br

MobDM – Jun 2016
Enrol Mail Dataset (http://www.enron-mail.com/)


Facebook Dataset

Contact Network Dataset

Temporal Networks