Information Diffusion at Twitter

Stanislav Nikolov
OVERVIEW

• Theory:
  – simple model of information diffusion

• Experiment:
  – diffusion of a topic on Twitter.
MOTIVATION

• How do things spread in a network?
  – Cascading failure (finance, power grids, etc.)
  – Opinion
  – Behavior
  – Innovations
  – Rumors

• Why do some things spread and others don’t?

• Can we detect (early) when something will spread?
OVERVIEW

• Theory:
  – simple model of information diffusion

• Experiment:
  – diffusion of a topic on Twitter.
TYPES OF DIFFUSION

• “Simple” vs. “complex”
• Simple – needs only one person
  – can happen randomly (e.g. disease)
• Complex – needs several people
  – Awareness (e.g. joke, event)
  – Credibility (e.g. rumor)
  – Legitimacy (e.g. occupy wall street)
  – Persuasiveness (e.g. “please RT”)
DIFFUSION AS DECISION-MAKING

• What’s a good decision rule?
DIFFUSION AS DECISION-MAKING

• What’s a good decision rule?
• Local information + threshold:

Figure 19.2: $v$ must choose between behavior $A$ and behavior $B$, based on what its neighbors are doing.

DIFFUSION AS DECISION-MAKING

• Game theoretic perspective – a networked coordination game

\[
\begin{array}{c|cc}
 & A & B \\
\hline
A & a, a & 0, 0 \\
B & 0, 0 & b, b \\
\end{array}
\]

Figure 19.1: A-B Coordination Game

• if \( v \) and \( w \) both adopt behavior \( A \), they each get a payoff of \( a > 0 \);
• if they both adopt \( B \), they each get a payoff of \( b > 0 \); and
• if they adopt opposite behaviors, they each get a payoff of 0.

DIFFUSION AS DECISION-MAKING

Figure 19.2: $v$ must choose between behavior $A$ and behavior $B$, based on what its neighbors are doing.

A is the better choice if

\[ pda \geq (1 - p)db, \]

\[ p \geq \frac{b}{a + b}. \]

\[ q = \frac{b}{a + b} \]
Figure 19.3: Starting with $v$ and $w$ as the initial adopters, and payoffs $a = 3$ and $b = 2$, the new behavior $A$ spreads to all nodes in two steps. Nodes adopting $A$ in a given step are drawn with dark borders; nodes adopting $B$ are drawn with light borders.
EXAMPLES

(c) After one step, two more nodes have adopted

(d) After a second step, everyone has adopted

Figure 19.3: Starting with $v$ and $w$ as the initial adopters, and payoffs $a = 3$ and $b = 2$, the new behavior $A$ spreads to all nodes in two steps. Nodes adopting $A$ in a given step are drawn with dark borders; nodes adopting $B$ are drawn with light borders.

$q = \frac{2}{5}$
EXAMPLES

Figure 19.4: A larger example.

EXAMPLES

(a) Two nodes are the initial adopters

EXAMPLES

(b) *The process ends after three steps*

THE ROLE OF CLUSTERS

• Clusters are obstacles to diffusion

• **Cluster of density \( p \):** set of nodes such that each node in the set has \( p \) or greater fraction of its neighbors in the set.

![Diagram of clusters](image)

Figure 19.6: A collection of four-node clusters, each of density 2/3.

THE ROLE OF CLUSTERS

Figure 19.7: Two clusters of density 2/3 in the network from Figure 19.4.

THE ROLE OF CLUSTERS

• Consider threshold $q$, and some initial participants.
• Claim:
  – diffusion will stop $\iff$ remaining network has a cluster of density greater than $1-q$
Figure 19.8: The spread of a new behavior, when nodes have threshold $q$, stops when it reaches a cluster of density greater than $(1 - q)$.
OVERVIEW

• Theory:
  – simple model of information diffusion

• Experiment:
  – diffusion of a topic on Twitter.
DATA COLLECTION

• Topics
  – that are unique and popular (trends)
  – that spread from person to person (hashtag memes/games, not events)

• Spreading mechanism
  – Retweet networks are not enough
  – Better to track who saw the topic from followees and then chose to participate (or not)
DATA COLLECTION

• 1. Collect topics.
• 2. Filter and tag Tweets.
• 3. From follow graph, get edges that Tweet authors participate in.
• 4. Filter edges such that parents come before children.
• Result: Collection of timestamped nodes and edges for each topic.
DATA COLLECTION

inactive parents of active nodes
active nodes
inactive children of active nodes
1. COLLECT TOPICS

• Pick a window of time.
• Find out what was trending in that window
• Record the first time it became trending (so that we can ignore it after that)
2. FILTER AND TAG TWEETS

-- Generate a regular expression to match tweets with any of the topics.
union_query = foreach topics generate CONCAT(CONCAT('.*\b',name),'\b.*');
union_query = foreach (group union_query all) generate JoinBagOr($1);

-- Filter the tweets, leaving only those with the chosen topics.
tweets = filter tweets by (text matches union_query.$0);

tweets = foreach tweets
    generate
    id,
    user_id,
    text,
    time,
    flatten(Topics(text, topics_group.$1)) as (topic, topic_start_ms, topic_end_ms);
3. EXTRACT PARTICIPATING EDGES

-- Get all edges in which an author is the follower
edges_l = foreach (join tweets by user_id, edges by source_id) generate
3. EXTRACT PARTICIPATING EDGES

-- Get all edges in which an author is the follower
edges_l = foreach (join tweets by user_id, edges by source_id) generate
    edges::source_id as source_id,
3. EXTRACT PARTICIPATING EDGES

-- Get all edges in which an author is the follower
edges_l = foreach (join tweets by user_id, edges by source_id) generate
    edges::source_id as source_id,
    edges::destination_id as destination_id,
3. EXTRACT PARTICIPATING EDGES

-- Get all edges in which an author is the follower
edges_l = foreach (join tweets by user_id, edges by source_id) generate
    edges::source_id as source_id,
    edges::destination_id as destination_id,
    tweets::id as tweet_id,
3. EXTRACT PARTICIPATING EDGES

-- Get all edges in which an author is the follower
edges_l = foreach (join tweets by user_id, edges by source_id) generate
  edges::source_id as source_id,
  edges::destination_id as destination_id,
  tweets::id as tweet_id,
  tweets::topic as topic,
3. EXTRACT PARTICIPATING EDGES

-- Get all edges in which an author is the follower
edges_l = foreach (join tweets by user_id, edges by source_id) generate
  edges::source_id as source_id,
  edges::destination_id as destination_id,
  tweets::id as tweet_id,
  tweets::topic as topic,
  tweets::topic_start_ms as topic_start_ms,
3. EXTRACT PARTICIPATING EDGES

-- Get all edges in which an author is the follower
edges_l = foreach (join tweets by user_id, edges by source_id) generate
  edges::source_id as source_id,
  edges::destination_id as destination_id,
  tweets::id as tweet_id,
  tweets::topic as topic,
  tweets::topic_start_ms as topic_start_ms,
  tweets::topic_end_ms as topic_end_ms,
3. EXTRACT PARTICIPATING EDGES

-- Get all edges in which an author is the follower

edges_l = foreach (join tweets by user_id, edges by source_id) generate
  edges::source_id as source_id,
  edges::destination_id as destination_id,
  tweets::id as tweet_id,
  tweets::topic as topic,
  tweets::topic_start_ms as topic_start_ms,
  tweets::topic_end_ms as topic_end_ms,
  tweets::time as source_tweet_time;
3. EXTRACT PARTICIPATING EDGES

-- Get all edges in which an author is the follower
edges_l = foreach (join tweets by user_id, edges by source_id) generate
  edges::source_id as source_id,
  edges::destination_id as destination_id,
  tweets::id as tweet_id,
  tweets::topic as topic,
  tweets::topic_start_ms as topic_start_ms,
  tweets::topic_end_ms as topic_end_ms,
  tweets::time as source_tweet_time;

-- Get all edges in which an author is the followee
edges_r = foreach (join tweets by user_id, edges by destination_id) generate
  edges::source_id as source_id,
  edges::destination_id as destination_id,
  tweets::id as tweet_id,
  tweets::topic as topic,
  tweets::topic_start_ms as topic_start_ms,
  tweets::topic_end_ms as topic_end_ms,
  tweets::time as destination_tweet_time;
3. EXTRACT PARTICIPATING EDGES

<table>
<thead>
<tr>
<th>user_id</th>
<th></th>
<th>source</th>
<th></th>
<th>destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>50</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>23</td>
<td>56</td>
<td></td>
</tr>
</tbody>
</table>

source | destination
---|---
1 | 29
3 | 1
3. EXTRACT PARTICIPATING EDGES

<table>
<thead>
<tr>
<th>user_id</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>...</td>
</tr>
<tr>
<td>4</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>source</th>
<th>destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>29</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>56</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>source</th>
<th>destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
3. EXTRACT PARTICIPATING EDGES

edges = foreach (join edges_l by (source_id, destination_id, topic) FULL,
        edges_r by (source_id, destination_id, topic)) generate
3. EXTRACT PARTICIPATING EDGES

edges = foreach (join edges_l by (source_id, destination_id, topic) FULL, 
    edges_r by (source_id, destination_id, topic)) generate 
    (edges_r::source_id IS NULL ? edges_l::source_id : edges_r::source_id) as 
    source_id,
3. EXTRACT PARTICIPATING EDGES

edges = foreach (join edges_l by (source_id, destination_id, topic) FULL, edges_r by (source_id, destination_id, topic)) generate
    (edges_r::source_id IS NULL ? edges_l::source_id : edges_r::source_id) as source_id,
    (edges_r::destination_id IS NULL ? edges_l::destination_id : edges_r::destination_id) as destination_id,
3. EXTRACT PARTICIPATING EDGES

edges = foreach (join edges_l by (source_id, destination_id, topic) FULL, edges_r by (source_id, destination_id, topic)) generate
    (edges_r::source_id IS NULL ? edges_l::source_id : edges_r::source_id) as source_id,
    (edges_r::destination_id IS NULL ? edges_l::destination_id : edges_r::destination_id) as destination_id,
    (edges_r::topic IS NULL ? edges_l::topic : edges_r::topic) as topic,
3. EXTRACT PARTICIPATING EDGES

edges = foreach (join edges_l by (source_id, destination_id, topic) FULL, 
edges_r by (source_id, destination_id, topic)) generate 
(edges_r::source_id IS NULL ? edges_l::source_id : edges_r::source_id) as 
source_id,
(edges_r::destination_id IS NULL ? edges_l::destination_id : 
edges_r::destination_id) as destination_id,
(edges_r::topic IS NULL ? edges_l::topic : edges_r::topic) as topic,
(edges_r::topic_start IS NULL ? edges_l::topic_start : 
edges_r::topic_start) as topic_start,
3. EXTRACT PARTICIPATING EDGES

edges = foreach (join edges_l by (source_id, destination_id, topic) FULL, edges_r by (source_id, destination_id, topic)) generate
(edges_r::source_id IS NULL ? edges_l::source_id : edges_r::source_id) as source_id,
(edges_r::destination_id IS NULL ? edges_l::destination_id : edges_r::destination_id) as destination_id,
(edges_r::topic IS NULL ? edges_l::topic : edges_r::topic) as topic,
(edges_r::topic_start IS NULL ? edges_l::topic_start : edges_r::topic_start) as topic_start,
(edges_r::topic_end IS NULL ? edges_l::topic_end : edges_r::topic_end) as topic_end,
3. EXTRACT PARTICIPATING EDGES

defines = foreach (join edges_l by (source_id, destination_id, topic) FULL, edges_r by (source_id, destination_id, topic)) generate
(edges_r::source_id IS NULL ? edges_l::source_id : edges_r::source_id) as source_id,
(edges_r::destination_id IS NULL ? edges_l::destination_id : edges_r::destination_id) as destination_id,
(edges_r::topic IS NULL ? edges_l::topic : edges_r::topic) as topic,
(edges_r::topic_start IS NULL ? edges_l::topic_start : edges_r::topic_start) as topic_start,
(edges_r::topic_end IS NULL ? edges_l::topic_end : edges_r::topic_end) as topic_end,
edges_l::source_tweet_id as source_tweet_id,
3. EXTRACT PARTICIPATING EDGES

edges = foreach (join edges_l by (source_id, destination_id, topic) FULL, edges_r by (source_id, destination_id, topic)) generate
  (edges_r::source_id IS NULL ? edges_l::source_id : edges_r::source_id) as source_id,
  (edges_r::destination_id IS NULL ? edges_l::destination_id : edges_r::destination_id) as destination_id,
  (edges_r::topic IS NULL ? edges_l::topic : edges_r::topic) as topic,
  (edges_r::topic_start IS NULL ? edges_l::topic_start : edges_r::topic_start) as topic_start,
  (edges_r::topic_end IS NULL ? edges_l::topic_end : edges_r::topic_end) as topic_end,
  edges_l::source_tweet_id as source_tweet_id,
  edges_r::destination_tweet_id as destination_tweet_id,
3. EXTRACT PARTICIPATING EDGES

edges = foreach (join edges_l by (source_id, destination_id, topic) FULL,
    edges_r by (source_id, destination_id, topic)) generate
    (edges_r::source_id IS NULL ? edges_l::source_id : edges_r::source_id) as source_id,
    (edges_r::destination_id IS NULL ? edges_l::destination_id :
    edges_r::destination_id) as destination_id,
    (edges_r::topic IS NULL ? edges_l::topic : edges_r::topic) as topic,
    (edges_r::topic_start IS NULL ? edges_l::topic_start :
    edges_r::topic_start) as topic_start,
    (edges_r::topic_end IS NULL ? edges_l::topic_end : edges_r::topic_end) as topic_end,
    edges_l::source_tweet_id as source_tweet_id,
    edges_r::destination_tweet_id as destination_tweet_id,
    edges_l::source_tweet_time as source_tweet_time,
edges = foreach (join edges_l by (source_id, destination_id, topic) FULL, edges_r by (source_id, destination_id, topic)) generate
(.edges_r::source_id IS NULL ? edges_l::source_id : edges_r::source_id) as source_id,
(edges_r::destination_id IS NULL ? edges_l::destination_id : edges_r::destination_id) as destination_id,
(edges_r::topic IS NULL ? edges_l::topic : edges_r::topic) as topic,
(edges_r::topic_start IS NULL ? edges_l::topic_start : edges_r::topic_start) as topic_start,
(edges_r::topic_end IS NULL ? edges_l::topic_end : edges_r::topic_end) as topic_end,
edges_l::source_tweet_id as source_tweet_id,
edges_r::destination_tweet_id as destination_tweet_id,
edges_l::source_tweet_time as source_tweet_time,
edges_r::destination_tweet_time as destination_tweet_time,
3. EXTRACT PARTICIPATING EDGES

edges = foreach (join edges_l by (source_id, destination_id, topic) FULL, edges_r by (source_id, destination_id, topic)) generate
(edges_r::source_id IS NULL ? edges_l::source_id : edges_r::source_id) as source_id,
(edges_r::destination_id IS NULL ? edges_l::destination_id : edges_r::destination_id) as destination_id,
(edges_r::topic IS NULL ? edges_l::topic : edges_r::topic) as topic,
(edges_r::topic_start IS NULL ? edges_l::topic_start : edges_r::topic_start) as topic_start,
(edges_r::topic_end IS NULL ? edges_l::topic_end : edges_r::topic_end) as topic_end,
edges_l::source_tweet_id as source_tweet_id,
edges_r::destination_tweet_id as destination_tweet_id,
edges_l::source_tweet_time as source_tweet_time,
edges_r::destination_tweet_time as destination_tweet_time,
(edges_r::source_id IS NULL ? 1 : (edges_l::source_id IS NULL ? -1 : 0)) as type;
3. EXTRACT PARTICIPATING EDGES

<table>
<thead>
<tr>
<th>source</th>
<th>destination</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>29</td>
<td>1</td>
</tr>
</tbody>
</table>
4. FILTER EDGES BY TIME

edges = filter edges by (source_tweet_time < topic_start);
DATA COLLECTION

- inactive parents of active nodes
- active nodes
- inactive children of active nodes
#DebateLinesForRomney
#BackStreetCruiseAlright
#GalinhaPintadinhaHumilhaLuaBlanco
#YouBelongOnJerrySpringer (up to 2 parents)
#TurkishDirectionersLovesHarryStayStrong
#AustraliaNeedsBelieveTourDates
linear-linear

![Graph with linear-linear scale](image-url)
log-linear
log-log
EXPERIMENTAL DIFFICULTIES

• Difficulty of choosing topics
  – Do topics stop spreading or start trending?
  – Look at rate of growth: is it slowing down or speeding up?
  – Look at candidates for trends that never become trends.

• Person-to-person vs. exogenous influence
  – How else is the topic being spread?

• Real users, followings are not all the same
  – Could estimate each person’s individual threshold
CONCLUSIONS

• Clusters are an obstacle to widespread diffusion (in theory)
• Tracking information spreading in a real network is non-trivial
• Simple models can be useful conceptually but tricky to apply to real data.
FUTURE WORK

• Better selection of topics
• Per user model
• More of graph around active nodes to study effect of clustering.
QUESTIONS?