DM-Group Meeting

Liangzhe Chen, Apr. 2 2015

Papers to be present

- On Integrating Network and Community Discovery
  - WSDM'15
  - J. Liu, C. Aggarwal, J. Han.
- Global Diffusion via Cascading Invitations: Structure, Growth and Homophily
  - WWW'15
  - A. Anderson, D. Huttenlocher, J. Kleigburg, J. Leskovec, M. Tiwari.

1st Paper

- On Integrating Network and Community Discovery
  - WSDM'15
  - J. Liu, C. Aggarwal, J. Han.

Introduction

- Most algorithms for community detection assume that the entire network is available for analysis.
  - Privacy constraints in Facebook
  - Hard to crawl the whole network in Twitter
  - Discovery of the entire network itself is a costly task
- Can we integrate community detection with network discovery?

Problem Definition

- G(N,A): N is the set of all nodes, A is the set of all edges in the network.
- $G_s(Ns,As,Qs)$: N_s is the set of observed nodes, A_s is the set of observed edges, Q_s are the costs to query nodes in N_s.
- **Given** $G_s(Ns,As,Qs)$, a target node set N_t (subset of N_s), an ability to query any currently observe node for their adjacent links at cost c_i, **cluster** N_t into the set of k most tightly linked communities within a total budget B.

Framework

Inialization

Get k clusters

Select a node to query,
And update the graph

Update the clusters

Algorithm *NetDiscover*(Initial Network: G_s, Budget: B, Number of Communities: K, Target Node set: N_t)

$G_c = G_s = (N_s, A_s, Q_s)$
Initialize current cost $q = 0$
Query target nodes N_t, update G_c, q
Initialize sampled nodes set $S = N_t$
$\mathcal{C}_c$ = Initial clustering on subgraph of G_c induced by S
while *budget* B *is not exhausted* **do**
  $i = ChooseNode(G_c, \mathcal{C}_c, S, q, B)$
  $S = S \cup \{i\}$
  $q = q + q_i$
  $G_i = (N_i, A_i, Q_i) = DiscoverLocality(i)$
  $G_c = G_c \cup G_i = (N_c \cup N_i, A_c \cup A_i, Q_c \cup Q_i)$
  $\mathcal{C}_c = UpdateCommunity(G_c, i, S, \mathcal{C}_c)$
end

Algorithm 1: Integrating Network and Community Discovery

How to select a node to query

Calculate a score for Each candidate

Adjust the score according to the cost

Function *ChooseNode*(Current Network: G_c, Clustering: $\mathcal{C}_c$, Sampled nodes set: S, Current cost: q, Budget: B)

Initialize dictionary D (key: candidate nodes, value: score)

for *each node* i *in* $N_c - S$ **do**
  if $q + q_i \leq B$ **then**
    $D[i] = ComputeScore(G_c, S, \mathcal{C}_c, i)$
  end
end

$AdjustScore(D, Q_c)$

if D *is empty* **then**
  Send Message: Budget B is exausted
else
  return node with lowest score in D
end

Function 1: Network Discovery

How to select a node to query

- Two ways used to calculate scores for nodes
  - Normalized cut

$$\sum_{k=1}^{K} \frac{cut(C_c^k, S - C_c^k)}{assoc(C_c^k, S)}$$

  - Modularity

$$\sum_{k=1}^{K} \left(e(C_c^k, S) - a(C_c^k, S)^2 \right)$$

How to select a node to query

- Incorporating the costs Q_c
  - For each node i, the rank of that node is adjusted by the cost of querying that node according to the following equation:

$$D[i] = rank(i) \times q_i^{\mu}$$

Parameter that controls how much the cost affect the result ranks

Community Discovery

- A generative model for the graph:
  - θ_{ik}: the propensity of a node i to have edges of community k
  - $\sum_k \theta_{ik}\theta_{jk}$: the expected number of links between node i and j
  - The likelihood of the graph:

$$P(G|\theta) = \prod_{i \le j} \frac{(\sum_k \theta_{ik}\theta_{jk})^{A_{ij}}}{A_{ij}!} \exp(-\sum_k \theta_{ik}\theta_{jk})$$

- Parameter updating rules (see details in the paper)

Recap of their algorithm

Inialization

Get k clusters

Select a node to query,
And update the graph

Update the clusters

Algorithm *NetDiscover*(Initial Network: G_s, Budget: B, Number of Communities: K, Target Node set: N_t)

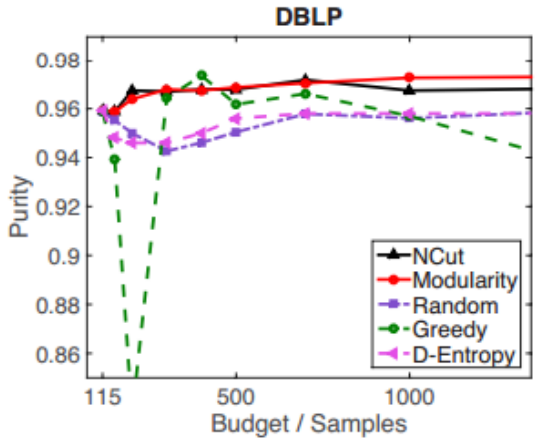
$G_c = G_s = (N_s, A_s, Q_s)$
Initialize current cost $q = 0$
Query target nodes N_t, update G_c, q
Initialize sampled nodes set $S = N_t$
$\mathcal{C}_c$ = Initial clustering on subgraph of G_c induced by S
while *budget* B *is not exhausted* **do**
  $i = ChooseNode(G_c, \mathcal{C}_c, S, q, B)$
  $S = S \cup \{i\}$
  $q = q + q_i$
  $G_i = (N_i, A_i, Q_i) = DiscoverLocality(i)$
  $G_c = G_c \cup G_i = (N_c \cup N_i, A_c \cup A_i, Q_c \cup Q_i)$
  $\mathcal{C}_c = UpdateCommunity(G_c, i, S, \mathcal{C}_c)$
end

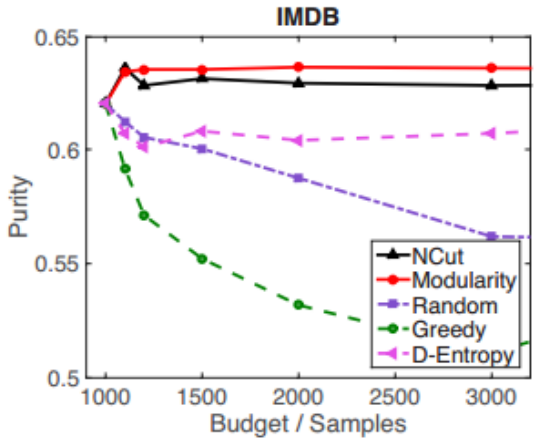
Algorithm 1: Integrating Network and Community Discovery

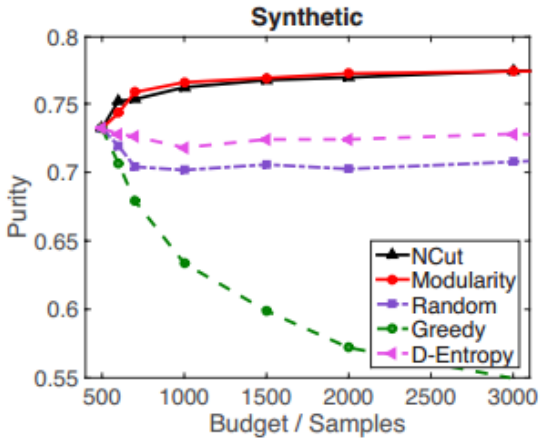
Experiments: Datasets

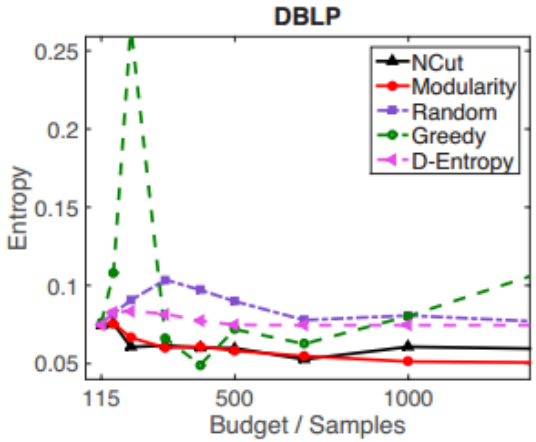
- Synthetic
  - 36,000 nodes, 6000 of them are generated from 5 clusters. Each of them has 3 out-cluster neighbors, and 8 within-cluster neighbors. The rest 30,000 nodes have random links.
- DBLP
  - Co-authorship network. 115 authors, from 4 research groups
- IMDB
  - Co-actor and co-director network. Different genres are treated as different clusters.

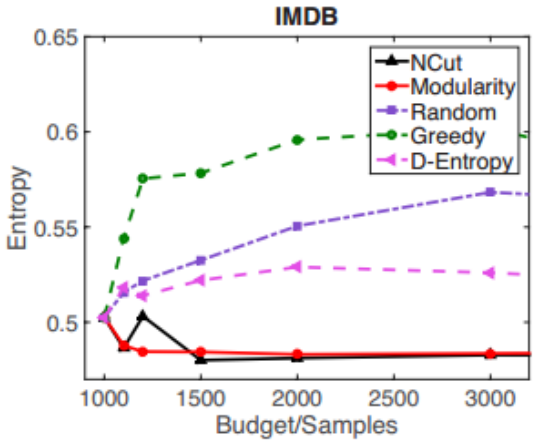
Experiments: Results

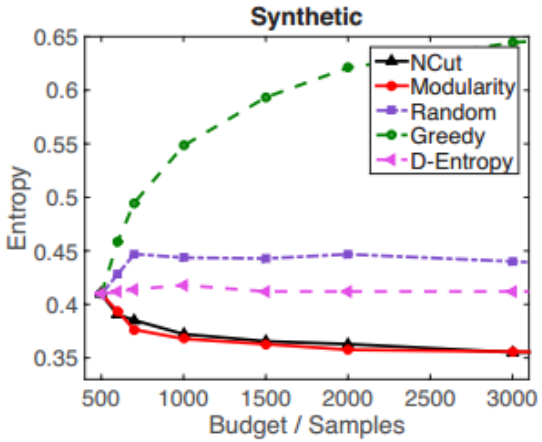












Experiments: Results

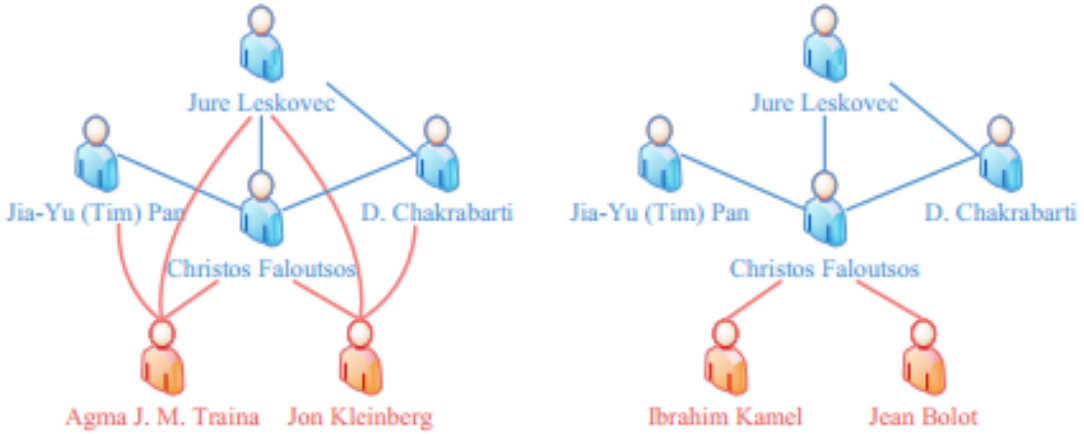
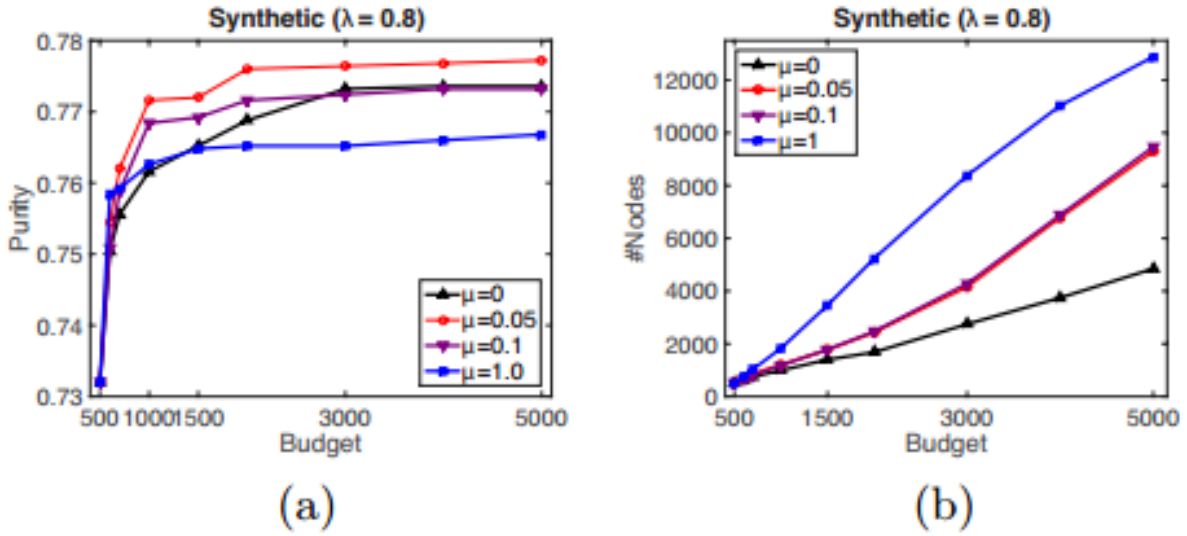


Figure 4: (a) Neighborhood of Prof. Christos Faloutsos using Ncut-based Search (b) using random sampling approach

Experiments: Results



(a) (b)

Figure 5: (a) Performance of *NetDiscover* with respect to budget (different μ) (b) Total discovery wrt budget (different μ)

2nd Papers

- Global Diffusion via Cascading Invitations: Structure, Growth and Homophily
  - WWW'15
  - A. Anderson, D. Huttenlocher, J. Kleigburg, J. Leskovec, M. Tiwari.

Introduction

- Many of the popular websites catalyze their growth through invitation from existing members. New members can then in turn issue invitations, thus creating a cascade of member signups.

Member Signups

- Two ways to sign up
  - A cold signup: sign up directly at the site
  - A warm signup: sign up through clicking an invitation from others
- Forming a graph of forest
  - Cold signups as root nodes
  - Ward signups have 1 parent

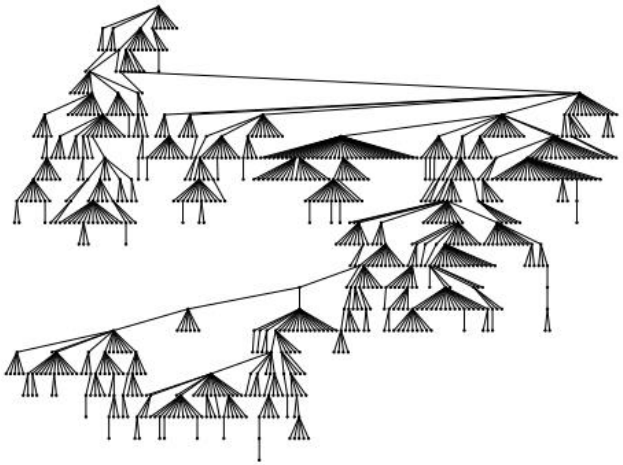


Figure 1: Example LinkedIn signup cascade.

Quantifying virality as a while

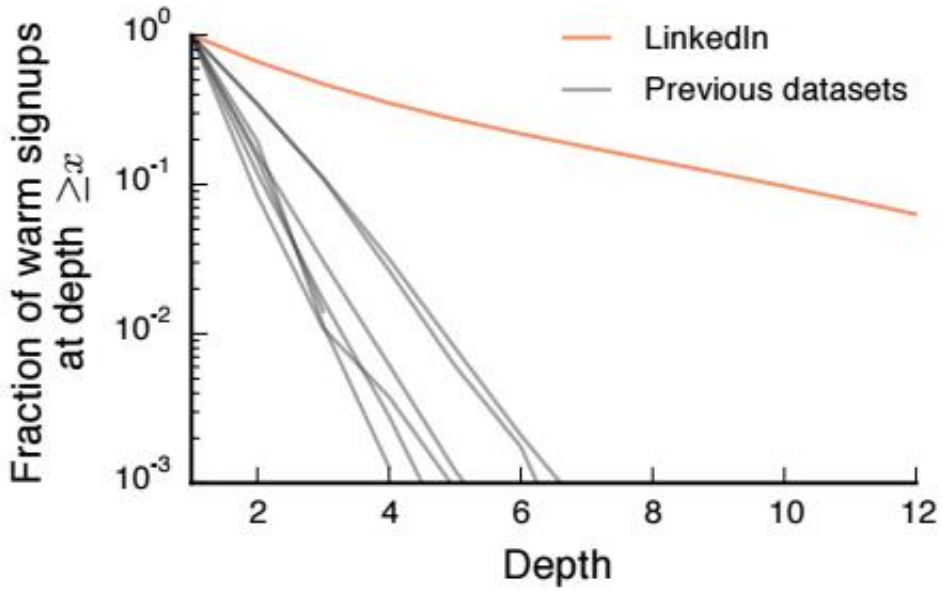


Figure 2: Distribution over adoption depth, excluding root nodes. LinkedIn adoptions occur much further from the root.

Quantifying virality as a while

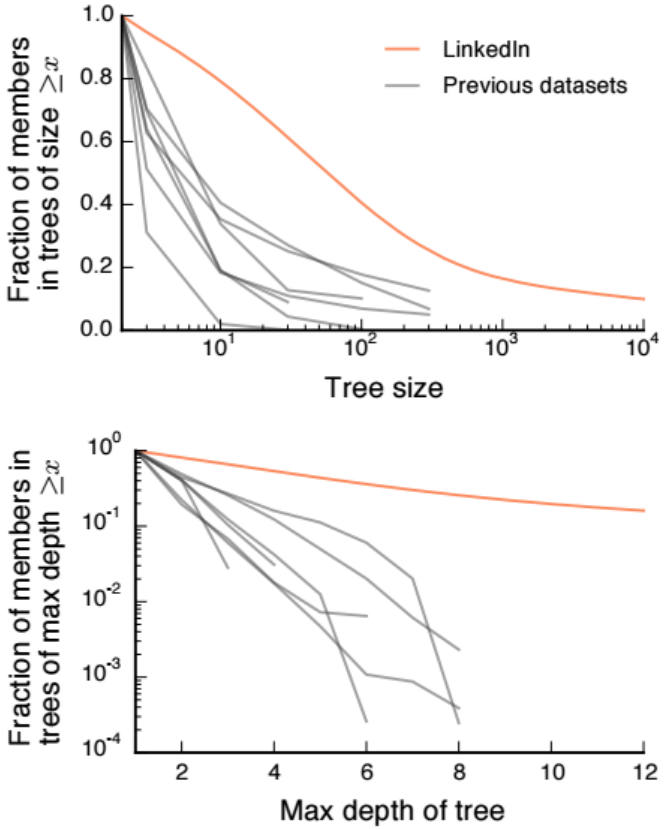


Figure 3: Fraction of non-singleton members in trees of specific size and depth. A greater portion of the LinkedIn signup forest is concentrated in large and deep cascades compared to previously studied diffusion datasets.

Structural Virality

- The goal of structural virality, is to numerically disambiguate between shallow broadcast like diffusions and the deep branching structures.
  - Use Wiener Index to capture the structural virality of a tree: average path distance between two nodes in the tree.

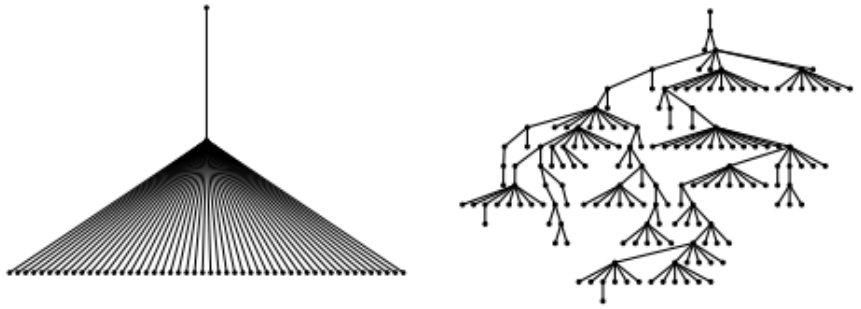


Figure 4: Two LinkedIn signup cascades, one with (left) low structural virality (Wiener index = 1.99), and one with (right) high structural virality (Wiener index = 9.5).

Structural Virality

- High correlation between cascade size and structural virality, different from other datasets.

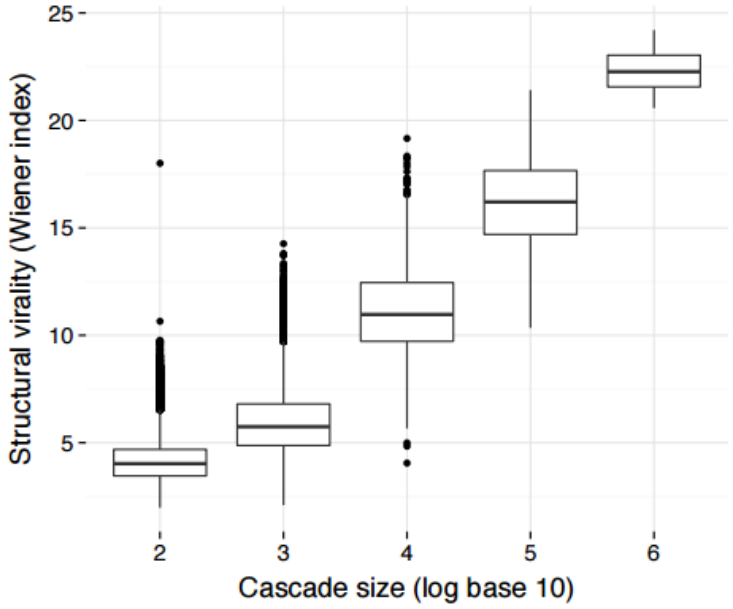


Figure 5: Structural virality as a function of cascade size (log base 10). The correlation is remarkably high, in contrast with previous findings on information-sharing cascades.

Homophily

- Edge homophily
- Cascade homophily

Edge Homophily

- Directly calculating $P(A_i|A_i)$
- High edge homophily is present in the dataset

Cascade Homophily

- Population diversity measure used in sociology
  - Within-similarity $W_A(T)$ of a group T on attribute A
    - Probability that two randomly selected nodes in T match on attribute A
  - Between-similarity $B_A(T_1,T_2)$
    - Probability that a randomly selected node in T_1 and a randomly selected node in T_2 match on attribute A
- Comparing W_A and B_A to identify cascade homophily.

Cascade Homophily

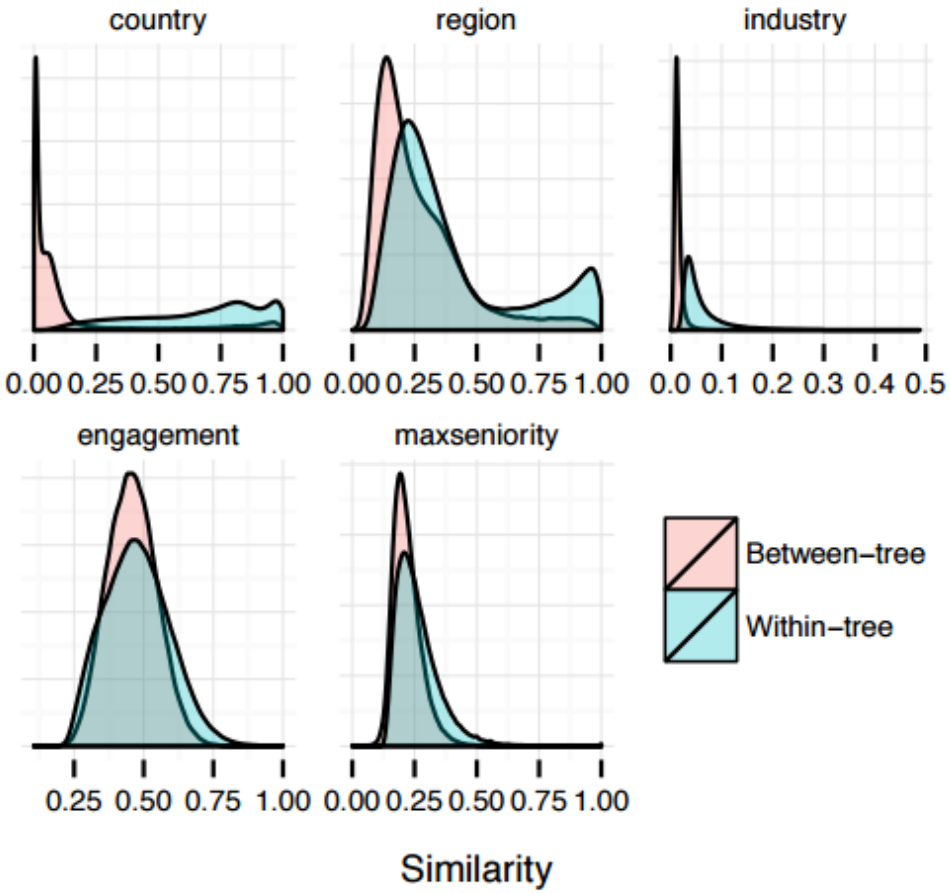


Figure 6: Within-tree and between-tree similarity on country, region, industry, engagement, and maximum job seniority.

Cascade Homophily

- Different attribute values show different level of homophily

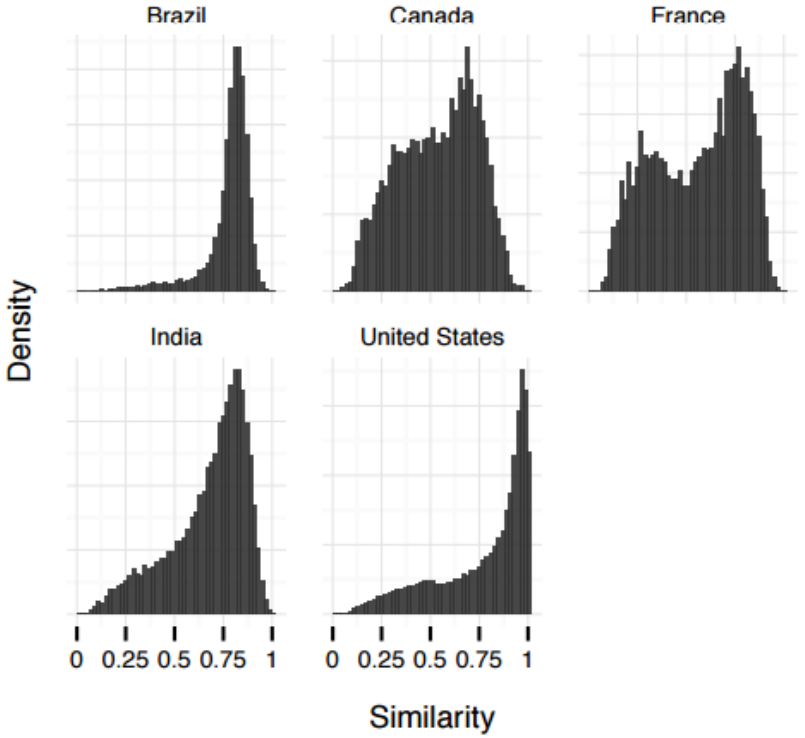


Figure 7: Within-tree similarity for trees rooted in Brazil, Canada, France, India, and the US.

Cascade & Edge Homophily

- Is the cascade homophily the same as the local edge homophily
  - Model the edge homophily by first order Markov chain using $P(A_i|A_j)$
  - Simulate the cascade tree using the Markov model and compare to the real tree.

Cascade & Edge Homophily

- First order Markov chain does not recover the data well.
  - The attributes of users are not entirely determined by the attributes of their direct parents, but by the rest of the cascade as well.
  - Edge level homophily is insufficient to explain cascade level homophily.

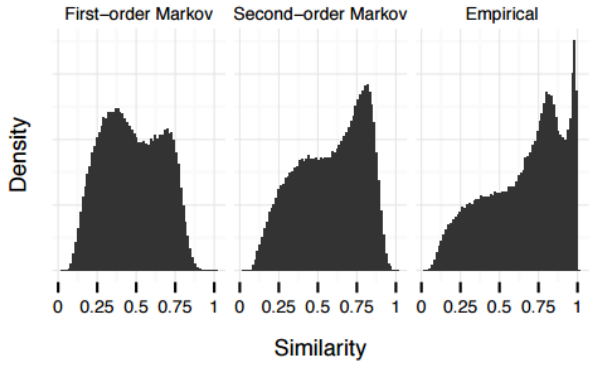


Figure 8: Within-tree similarity on real tree topologies with countries drawn from: (left) first-order Markov transitions M_1, and (middle) second-order Markov transitions M_2; (right) empirical within-tree similarity.

Guessing the root

- The edge homophily suggests that the cascade tends to retain some memory of the root. How quickly the cascade lose its root information and relax to the background distribution?

Guessing the root

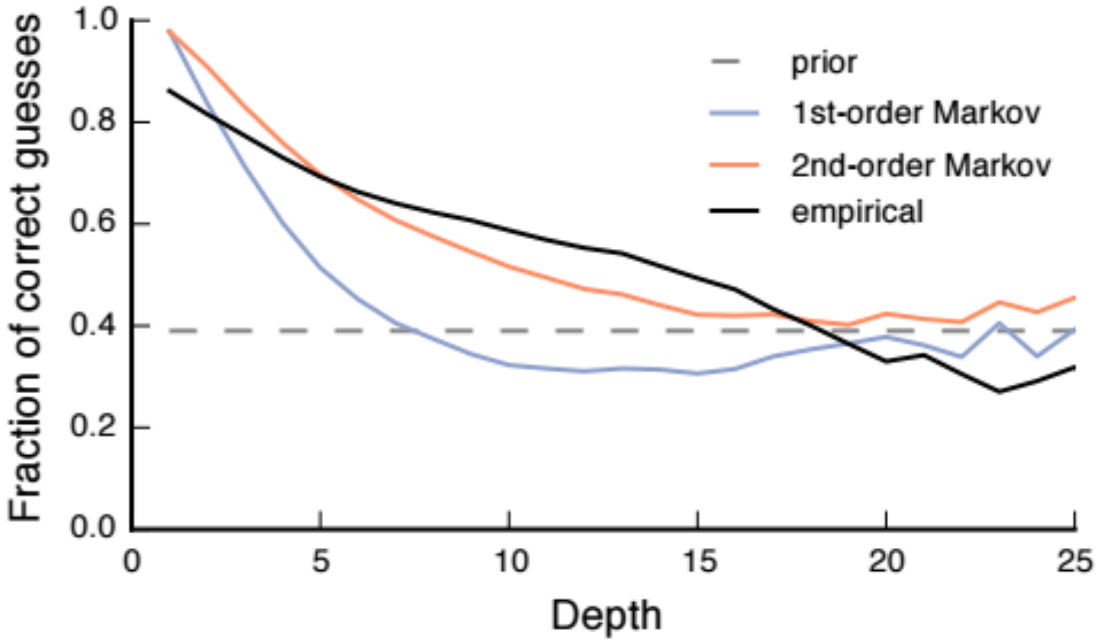
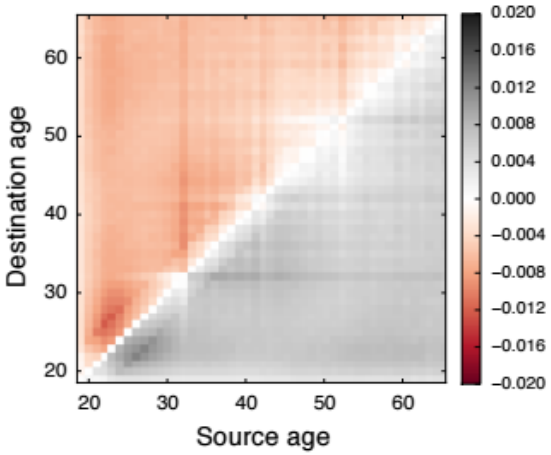
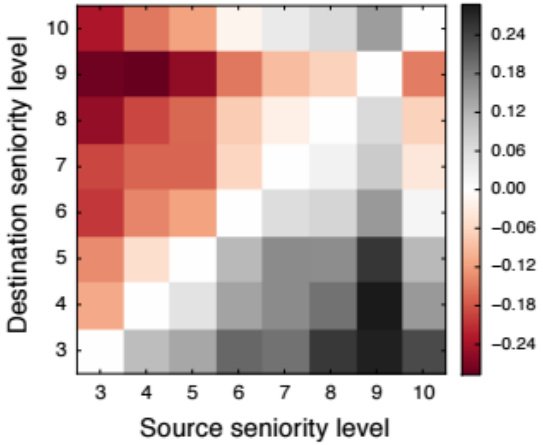


Figure 9: Fraction of time plurality attribute at depth d matches root attribute in root-guessing experiment. Empirical data retains "memory" of the root longer than baselines.

Status Gradient





- Status gradient is observed in some of the attributes which do not show homophily

Timescale of transmission

- Invitations to others are sent long after the registration of the user.
- Invitations are adopted quickly after a user receives one.

Cascade Growth Trajectories

- Cascade size grows almost linearly w.r.t time.