UNIVERSITY OF TWENTE.

Quick detection of popular entities in large on-line networks

Nelly Litvak

University of Twente,
Stochastic Operations Research group

Joint work with
K. Avrachenkov (INRIA), L. Ostroumova (Yandex)

Luchon 24-06-2014

Finding largest nodes in large complex networks

- Complex networks: Internet, World Wide Web, social networks, protein-protein interactions, citation networks.

Finding largest nodes in large complex networks

- Complex networks: Internet, World Wide Web, social networks, protein-protein interactions, citation networks.
- Many networks are very large.

Finding largest nodes in large complex networks

- Complex networks: Internet, World Wide Web, social networks, protein-protein interactions, citation networks.
- Many networks are very large.

- Facebook has more than 1 billion users. With an average user having 190 friends, the number of social links in Facebook is 190 billion.

- The static part of the web graph has more than 10 billion pages. With an average number of 38 hyper-links per page, the total number of hyper-links is 380 billion.

Finding top-k largest degree nodes

- Goal: Find top-k network nodes with largest degrees

Finding top-k largest degree nodes

- Goal: Find top-k network nodes with largest degrees
- Some applications:
  - Routing via large degree nodes
  - Proxy for various centrality measures
  - Node clustering and classification
  - Epidemic processes on networks
  - Finding most popular entities (e.g. interest groups)

Finding top-k largest degree nodes

- Goal: Find top-k network nodes with largest degrees
- Some applications:
  - Routing via large degree nodes
  - Proxy for various centrality measures
  - Node clustering and classification
  - Epidemic processes on networks
  - Finding most popular entities (e.g. interest groups)
  - It is simply interesting!

Top-k largest degree nodes

If the adjacency list of the network is known...

the top-k list of nodes can be found by the HeapSort with complexity $O(N + klog(N))$, where N is the total number of nodes.

Even this modest complexity can be demanding for large networks.

Top-k largest degree nodes

If the adjacency list of the network is known...

the top-k list of nodes can be found by the HeapSort with complexity $O(N + klog(N))$, where N is the total number of nodes.

Even this modest complexity can be demanding for large networks.

Questions:

- How to do this faster?

Top-k largest degree nodes

If the adjacency list of the network is known...

the top-k list of nodes can be found by the HeapSort with complexity $O(N + klog(N))$, where N is the total number of nodes.

Even this modest complexity can be demanding for large networks.

Questions:

- How to do this faster?
- How to do it when the network structure is not known (cannot be crawled without restrictions or stored in the memory)?

Top-k largest degree nodes

If the adjacency list of the network is known...

the top-k list of nodes can be found by the HeapSort with complexity $O(N + klog(N))$, where N is the total number of nodes.

Even this modest complexity can be demanding for large networks.

Questions:

- How to do this faster?
- How to do it when the network structure is not known (cannot be crawled without restrictions or stored in the memory)?

Answer: Randomized algorithms.

Top-k largest degree nodes

If the adjacency list of the network is known...

the top-k list of nodes can be found by the HeapSort with complexity $O(N + klog(N))$, where N is the total number of nodes.

Even this modest complexity can be demanding for large networks.

Questions:

- How to do this faster?
- How to do it when the network structure is not known (cannot be crawled without restrictions or stored in the memory)?

Answer: Randomized algorithms.
Idea: Find a 'good enough' answer in a short time.

Top-k largest degree nodes

If the adjacency list of the network is known...

the top-k list of nodes can be found by the HeapSort with complexity $O(N + klog(N))$, where N is the total number of nodes.

Even this modest complexity can be demanding for large networks.

Questions:

- How to do this faster?
- How to do it when the network structure is not known (cannot be crawled without restrictions or stored in the memory)?

Answer: Randomized algorithms.

Idea: Find a 'good enough' answer in a short time.

Avrachenkov, L, Sokol, Towsley (2012); Cooper, Radzik, Siantos (2012), Borgs, Brautbar, Chayes, Khanna, Lucier (2012),
Brautbar and Kearns (2010), Kumar, Lang, Marlow, Tomkins (2008)

Finding most popular entities in directed on-line social networks

- Social networks are large

Finding most popular entities in directed on-line social networks

- Social networks are large
- The complete graphs structure is only available to the owners

Finding most popular entities in directed on-line social networks

- Social networks are large
- The complete graphs structure is only available to the owners
- Many companies maintain network statistics (*twittercounter.com*, *followerwonk.com*, *twitaholic.com*, *www.insidefacebook.com*, *yavkontakte.ru*)

Finding most popular entities in directed on-line social networks

- Social networks are large
- The complete graphs structure is only available to the owners
- Many companies maintain network statistics (*twittercounter.com*, *followerwonk.com*, *twitaholic.com*, *www.insidefacebook.com*, *yavkontakte.ru*)
- The network can be accessed only via API, with limited access

Finding most popular entities in directed on-line social networks

- Social networks are large
- The complete graphs structure is only available to the owners
- Many companies maintain network statistics (*twittercounter.com*, *followerwonk.com*, *twitaholic.com*, *www.insidefacebook.com*, *yavkontakte.ru*)
- The network can be accessed only via API, with limited access
- Twitter API allows one access per minute. We need 950 years to crawl the current Twitter graph!

Finding most popular entities in directed on-line social networks

- Social networks are large
- The complete graphs structure is only available to the owners
- Many companies maintain network statistics (*twittercounter.com*, *followerwonk.com*, *twitaholic.com*, *www.insidefacebook.com*, *yavkontakte.ru*)
- The network can be accessed only via API, with limited access
- Twitter API allows one access per minute. We need 950 years to crawl the current Twitter graph!

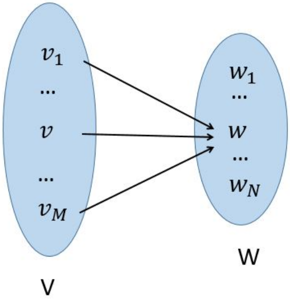
Goal: Find top-k most popular entities in social (directed) networks (nodes with highest in/out-degrees, largest interest groups, largest user categories), using the minimal number of API requests.

Problem formulation

- Consider a bi-partite graph (V, W, E)
- V and W are sets of entities, $|V| = M$, $|W| = N$.
- A directed edge $(v, w) \in E$ represents a relation between $v \in V$ and $w \in W$.
- Goal: Quickly find entities in W with highest degrees.

Problem formulation

- Consider a bi-partite graph (V, W, E)
- V and W are sets of entities, $|V| = M$, $|W| = N$.
- A directed edge $(v, w) \in E$ represents a relation between $v \in V$ and $w \in W$.
- Goal: Quickly find entities in W with highest degrees.



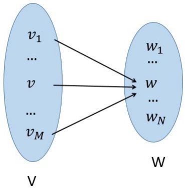
Example. $V = W$ is a set of Twitter users, (v, w) means that v follows w.

Example. V is a set of users, W is a set of interest groups, (v, w) means that user v is a member of an interest group w.

Algorithm for finding top-k most popular entities

Algorithm for finding top-k most popular entities

1. Choose a set $A \subset V$ of n_1 nodes sampled from V at random.
2. For each $v \in A$ retrieve the id's of nodes in W that have an edge from v.
3. Compute S_w – the number of edges of $w \in W$ from A.
4. Retrieve the actual degrees for the n_2 nodes w with the largest values of S_w.
5. Return the identified top-k list of most popular entities in W.



In total, we use $n = n_1 + n_2$ requests to API (Step 2 and Step 4).

Finding most followed users on Twitter

- Huge network (more than 500M users)

Finding most followed users on Twitter

- Huge network (more than 500M users)
- Network accessed only through Twitter API

Finding most followed users on Twitter

- Huge network (more than 500M users)
- Network accessed only through Twitter API
- The rate of requests is limited
- One request:
  - ID's of at most 5000 followers of a node, or
  - the number of followers of a node
- In a randomly chosen set of n_1 Twitter users only a few users follow more than 5000 people. Thus, we retrieve at most 5000 followees of each node. This does not affect the results.

Finding most followed users on Twitter

- Huge network (more than 500M users)
- Network accessed only through Twitter API
- The rate of requests is limited
- One request:
  - ID's of at most 5000 followers of a node, or
  - the number of followers of a node
- In a randomly chosen set of n_1 Twitter users only a few users follow more than 5000 people. Thus, we retrieve at most 5000 followees of each node. This does not affect the results.
- Make a guess: We use 1000 requests to API. For which k can we identify a top-k list of most followed Twitter users with 90% precision?

Results

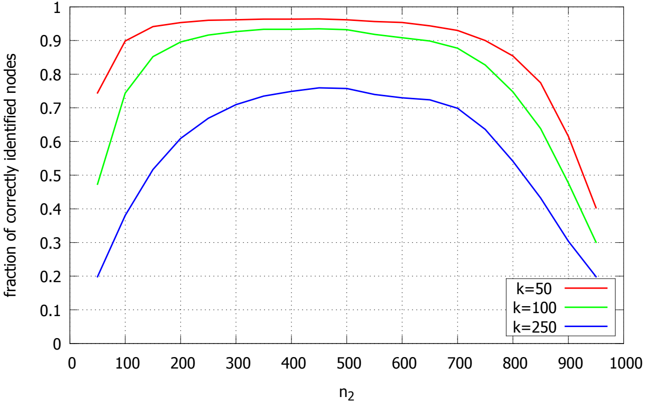


Figure : The fraction of correctly identified top-k most followed Twitter users as a function of n_2, with $n = 1000$.

Most followed

Twitter users		Followers	Following	Tweets
1	KATY PERRY @katyperry	53,923,965	148	5,699
2	Justin Bieber @justinbieber	52,445,383	130,204	27,064
3	Barack Obama @BarackObama	43,712,727	650,033	11,955
4	YouTube @YouTube	43,007,224	704	10,599
5	Lady Gaga @ladygaga	41,548,506	134,424	4,782

Interest groups VKontakte

- Popular social network in Russian, more than 200M users.

Rank	Number of participants	Topic
1	4,35M	humor
2	4,1M	humor
3	3,76M	movies
4	3,69M	humor
5	3,59M	humor
6	3,58M	facts
7	3,36M	cookery
8	3,31M	humor
9	3,14M	humor
10	3,14M	movies
100	1,65M	success

- With $n_1 = 700$, $n_2 = 300$, our algorithm identifies on average 73.2 from the top-100 interest groups (averaged over 25 experiments). The standard deviation is 4.6.

Comparison to known algorithms

- Well-studied problem

Comparison to known algorithms

- Well-studied problem
- How our algorithm compares to baselines?

Algorithm by Cooper, Radzik, Siantos (2012)

- Random-walk based
- Transitions probabilities along undirected edges (x, y) are proportional to $(d(x)d(y))^b$, where $d(x)$ is the degree of a vertex x and $b > 0$ is some parameter.

Problems?

Algorithm by Cooper, Radzik, Siantos (2012)

- Random-walk based
- Transitions probabilities along undirected edges (x, y) are proportional to $(d(x)d(y))^b$, where $d(x)$ is the degree of a vertex x and $b > 0$ is some parameter.

Problems?

- Designed for undirected and connected graphs (preferential attachment graphs)

Algorithm by Cooper, Radzik, Siantos (2012)

- Random-walk based
- Transitions probabilities along undirected edges (x, y) are proportional to $(d(x)d(y))^b$, where $d(x)$ is the degree of a vertex x and $b > 0$ is some parameter.

Problems?

- Designed for undirected and connected graphs (preferential attachment graphs)
- We need $d(x)$ API requests to know the $d(y)$'s. All these resources are spent to make just ONE transition!

Algorithm by Cooper, Radzik, Siantos (2012)

- Random-walk based
- Transitions probabilities along undirected edges (x, y) are proportional to $(d(x)d(y))^b$, where $d(x)$ is the degree of a vertex x and $b > 0$ is some parameter.

Problems?

- Designed for undirected and connected graphs (preferential attachment graphs)
- We need $d(x)$ API requests to know the $d(y)$'s. All these resources are spent to make just ONE transition!
- Not implementable on Twitter

Random Walk

Avrachenkov, L, Sokol, Towsley (2012)

- Random walk with uniform jumps:

$$p(x,y) = \begin{cases} \frac{\alpha/N+1}{d(x)+\alpha}, & \text{if } x \text{ has a link to } y, \\ \frac{\alpha/N}{d(x)+\alpha}, & \text{if } x \text{ does not have a link to } y, \end{cases}$$

  where N is the number of nodes in the graph and $d(x)$ is the degree of a node x.
- Rationale: in undirected graphs the stationary distribution is given by

$$\pi_x(\alpha) = \frac{d(x)+\alpha}{2|E|+N\alpha}.$$

Random Walk

Avrachenkov, L, Sokol, Towsley (2012)

- Random walk with uniform jumps:

$$p(x,y) = \begin{cases} \frac{\alpha/N+1}{d(x)+\alpha}, & \text{if } x \text{ has a link to } y, \\ \frac{\alpha/N}{d(x)+\alpha}, & \text{if } x \text{ does not have a link to } y, \end{cases}$$

  where N is the number of nodes in the graph and $d(x)$ is the degree of a node x.
- Rationale: in undirected graphs the stationary distribution is given by

$$\pi_x(\alpha) = \frac{d(x)+\alpha}{2|E|+N\alpha}.$$

- Best to take α approximately equal to the average degree

Problems?

Random Walk: problems

- Undirected graphs:

$$\pi_x(\alpha) = \frac{d(x) + \alpha}{2|E| + N\alpha}.$$

In directed graphs, stationary distribution will not give the order according to degrees.

Random Walk: problems

- Undirected graphs:

$$\pi_x(\alpha) = \frac{d(x) + \alpha}{2|E| + N\alpha}.$$

  In directed graphs, stationary distribution will not give the order according to degrees.
- Fix: make the graph undirected (symmetrized). Usually in-degrees are larger than out-degrees, so ordering by total degree and by in-degree should be similar.

Random Walk: problems

- Undirected graphs:

$$\pi_x(\alpha) = \frac{d(x) + \alpha}{2|E| + N\alpha}.$$

  In directed graphs, stationary distribution will not give the order according to degrees.
- Fix: make the graph undirected (symmetrized). Usually in-degrees are larger than out-degrees, so ordering by total degree and by in-degree should be similar.

More problems?

Random Walk: problems

- Undirected graphs:

$$\pi_x(\alpha) = \frac{d(x) + \alpha}{2|E| + N\alpha}.$$

  In directed graphs, stationary distribution will not give the order according to degrees.

- Fix: make the graph undirected (symmetrized). Usually in-degrees are larger than out-degrees, so ordering by total degree and by in-degree should be similar.

More problems?

- We need to know ids of all neighbors of x to decide where to go, but we can obtain only 5000 ids per API request.

Random Walk: problems

- Undirected graphs:

$$\pi_x(\alpha) = \frac{d(x) + \alpha}{2|E| + N\alpha}.$$

  In directed graphs, stationary distribution will not give the order according to degrees.
- Fix: make the graph undirected (symmetrized). Usually in-degrees are larger than out-degrees, so ordering by total degree and by in-degree should be similar.

More problems?

- We need to know ids of all neighbors of x to decide where to go, but we can obtain only 5000 ids per API request.
- Strict: [one step of the algorithm] = [one API request]
- Relaxed: [one step of the algorithm] = [one considered vertex]

Crawl-AI and Crawl-GAI

Kumar, Lang, Marlow, Tomkins (2008)

- Designed for WWW crawl

Crawl-AI and Crawl-GAI

Kumar, Lang, Marlow, Tomkins (2008)

- Designed for WWW crawl
- At every step all nodes have their *apparent in-degrees* S_j, $j = 1, \ldots, N$: the number of discovered edges pointing to this node.
- Crawl-AI: the next node is chosen at random with probability proportional to its apparent in-degree
- Crawl-GAI: the next node is the node with the highest apparent in-degree

Crawl-AI and Crawl-GAI

Kumar, Lang, Marlow, Tomkins (2008)

- Designed for WWW crawl
- At every step all nodes have their *apparent in-degrees* S_j, $j = 1, \ldots, N$: the number of discovered edges pointing to this node.
- Crawl-AI: the next node is chosen at random with probability proportional to its apparent in-degree
- Crawl-GAI: the next node is the node with the highest apparent in-degree

Problems?

Crawl-AI and Crawl-GAI

Kumar, Lang, Marlow, Tomkins (2008)

- Designed for WWW crawl
- At every step all nodes have their *apparent in-degrees* S_j, $j = 1, \ldots, N$: the number of discovered edges pointing to this node.
- Crawl-AI: the next node is chosen at random with probability proportional to its apparent in-degree
- Crawl-GAI: the next node is the node with the highest apparent in-degree

Problems?

- The resulting list is created according to the *apparent in-degrees*, a lot of randomness

Crawl-AI and Crawl-GAI

Kumar, Lang, Marlow, Tomkins (2008)

- Designed for WWW crawl
- At every step all nodes have their *apparent in-degrees* S_j, $j = 1, \ldots, N$: the number of discovered edges pointing to this node.
- Crawl-AI: the next node is chosen at random with probability proportional to its apparent in-degree
- Crawl-GAI: the next node is the node with the highest apparent in-degree

Problems?

- The resulting list is created according to the *apparent in-degrees*, a lot of randomness
- Crawl-GAI can get stuck in some densely connected cluster

Crawl-AI and Crawl-GAI

Kumar, Lang, Marlow, Tomkins (2008)

- Designed for WWW crawl
- At every step all nodes have their *apparent in-degrees* S_j, $j = 1, \ldots, N$: the number of discovered edges pointing to this node.
- Crawl-AI: the next node is chosen at random with probability proportional to its apparent in-degree
- Crawl-GAI: the next node is the node with the highest apparent in-degree

Problems?

- The resulting list is created according to the *apparent in-degrees*, a lot of randomness
- Crawl-GAI can get stuck in some densely connected cluster
- Can suffer from correlations between in- and out-degrees

HighestDegree

Borgs, Brautbar, Chayes, Khanna, Lucier (2012)

- Retrieve a random node
- Check in-degrees of its out-neighbors
- Proceed while resources are available

HighestDegree

Borgs, Brautbar, Chayes, Khanna, Lucier (2012)

- Retrieve a random node
- Check in-degrees of its out-neighbors
- Proceed while resources are available

Problems?

HighestDegree

Borgs, Brautbar, Chayes, Khanna, Lucier (2012)

- Retrieve a random node
- Check in-degrees of its out-neighbors
- Proceed while resources are available

Problems?

- A lot of resources are spent on out-neighbors of random nodes

Comparison of the algorithms

Table : Percentage of correctly identified nodes from top-100 in Twitter averaged over 30 experiments, $n = 1000$

Algorithm	mean	standard deviation
Two-stage algorithm	92.6	4.7
Random walk (strict)	0.43	0.63
Random walk (relaxed)	8.7	2.4
Crawl-GAI	4.1	5.9
Crawl-AI	23.9	20.2
HighestDegree	24.7	11.8

Comparison of the algorithms

Table : Percentage of correctly identified nodes from top-100 in Twitter averaged over 30 experiments, $n = 1000$

Algorithm	mean	standard deviation
Two-stage algorithm	92.6	4.7
Random walk (strict)	0.43	0.63
Random walk (relaxed)	8.7	2.4
Crawl-GAI	4.1	5.9
Crawl-AI	23.9	20.2
HighestDegree	24.7	11.8

Advantages of the two-stage algorithm:

- does not waste resources
- obtains *exact* degrees of the n_2 'most promising' nodes

Comparison of the algorithms

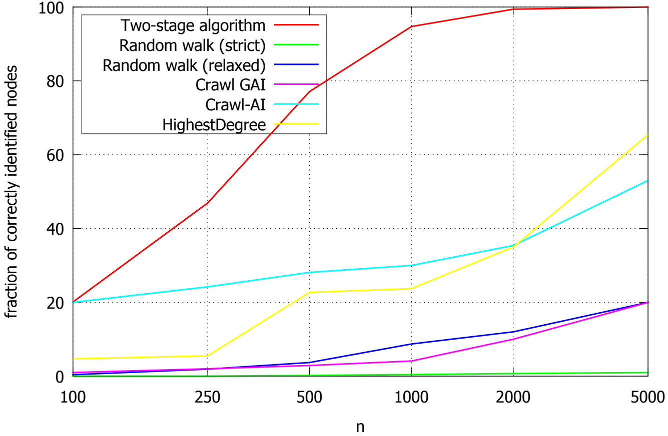


Figure : The fraction of correctly identified top-100 most followed Twitter users as a function of *n* averaged over 10 experiments.

Influence of graph size?

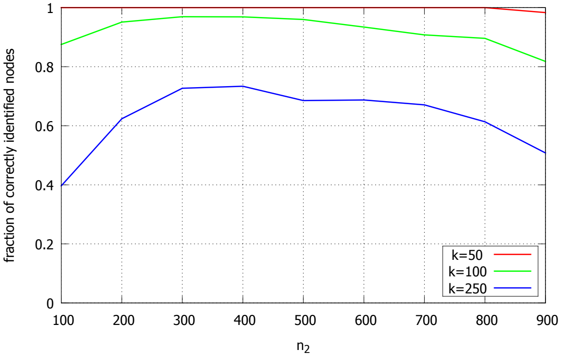


Figure : The fraction of correctly identified top-k in-degree nodes in the CNR-2000 graph (*law.di.unimi.it/webdata/cnr-2000*) as a function of n_2, with $n = 1000$. Note that algorithm performs similarly on CNR-2000 (half a million nodes) and Twitter.

Hubs in complex networks

- degree of the node = # links, [fraction nodes degree k] = p_k,

Hubs in complex networks

- degree of the node = # links, [fraction nodes degree k] = p_k,
- Power law: $p_k \approx const \cdot k^{-\gamma-1}$, $\gamma > 1$.

Hubs in complex networks

- degree of the node = # links, [fraction nodes degree k] = p_k,
- Power law: $p_k \approx const \cdot k^{-\gamma-1}$, $\gamma > 1$.
- Model for high variability, scale-free graph.
- Hubs are the nodes with extremely large degrees.

Hubs in complex networks

- degree of the node = # links, [fraction nodes degree k] = p_k,
- Power law: $p_k \approx const \cdot k^{-\gamma-1}$, $\gamma > 1$.
- Model for high variability, scale-free graph.
- Hubs are the nodes with extremely large degrees.

	Twitter users		Followers	Following	Tweets
1	KATY PERRY	@katyperry	53,923,965	148	5,699
2	Justin Bieber	@justinbieber	52,445,383	130,204	27,064
3	Barack Obama	@BarackObama	43,712,727	650,033	11,955
4	YouTube	@YouTube	43,007,224	704	10,599
5	Lady Gaga	@ladygaga	41,548,506	134,424	4,782

Formal view on the hubs

Let D be a degree of a random node. Regular varying distribution:

$$P(D > x) = L(x)x^{-\gamma} \tag{1}$$

$L(x)$ is slowly varying, i.e. $\lim_{t\to\infty} L(tx)/L(t) = 1$, $x \geqslant 0$

Formal view on the hubs

Let D be a degree of a random node. Regular varying distribution:

$$P(D > x) = L(x)x^{-\gamma} \tag{1}$$

$L(x)$ is slowly varying, i.e. $\lim_{t\to\infty} L(tx)/L(t) = 1$, $x \geqslant 0$

EXTREME VALUE THEORY. Let $F_1 \geqslant F_2 \geqslant \cdots \geqslant F_N$ be the order statistics of the i.i.d. r.v.'s $D_1, D_2, \ldots, D_N$ as in (1). Then there are (a_N) such that for finite k

$$\left(\frac{F_1}{a_N}, \cdots, \frac{F_k}{a_N}\right) \xrightarrow{d} \left(\frac{E_1^{-\delta}}{\delta}, \cdots, \frac{\left(\sum_{i=1}^{k} E_i\right)^{-\delta}}{\delta}\right),$$

where $\delta = 1/\gamma$ and E_i'are i.i.d. exponential(1) r.v.'s. k

Formal view on the hubs

Let D be a degree of a random node. Regular varying distribution:

$$P(D > x) = L(x)x^{-\gamma} \tag{1}$$

$L(x)$ is slowly varying, i.e. $\lim_{t\to\infty} L(tx)/L(t) = 1$, $x \geqslant 0$

EXTREME VALUE THEORY. Let $F_1 \geqslant F_2 \geqslant \cdots \geqslant F_N$ be the order statistics of the i.i.d. r.v.'s $D_1, D_2, \ldots, D_N$ as in (1). Then there are (a_N) such that for finite k

$$\left(\frac{F_1}{a_N}, \cdots, \frac{F_k}{a_N}\right) \xrightarrow{d} \left(\frac{E_1^{-\delta}}{\delta}, \cdots, \frac{\left(\sum_{i=1}^{k} E_i\right)^{-\delta}}{\delta}\right),$$

where $\delta = 1/\gamma$ and E_i'are i.i.d. exponential(1) r.v.'s. k

Example. $P(D > x) = Cx^{-\gamma}$, then $a_N = \delta C^{\delta} N^{\delta}$, $b_N = C^{\delta} N^{\delta}$.
The largest degrees are 'of the order' $N^{1/\gamma}$.

Formal view on the hubs

Let D be a degree of a random node. Regular varying distribution:

$$P(D > x) = L(x)x^{-\gamma} \quad (1)$$

$L(x)$ is slowly varying, i.e. $\lim_{t\to\infty} L(tx)/L(t) = 1$, $x \geqslant 0$

EXTREME VALUE THEORY. Let $F_1 \geqslant F_2 \geqslant \cdots \geqslant F_N$ be the order statistics of the i.i.d. r.v.'s $D_1, D_2, \ldots, D_N$ as in (1). Then there are (a_N) such that for finite k

$$\left(\frac{F_1}{a_N}, \cdots, \frac{F_k}{a_N}\right) \xrightarrow{d} \left(\frac{E_1^{-\delta}}{\delta}, \cdots, \frac{\left(\sum_{i=1}^{k} E_i\right)^{-\delta}}{\delta}\right),$$

where $\delta = 1/\gamma$ and E_i'are i.i.d. exponential(1) r.v.'s. k

Example. $P(D > x) = Cx^{-\gamma}$, then $a_N = \delta C^\delta N^\delta$, $b_N = C^\delta N^\delta$.
The largest degrees are 'of the order' $N^{1/\gamma}$.

Performance prediction

- Number nodes in W in the decreasing order of their degrees: $F_1 \geqslant F_2 \geqslant \cdots \geqslant F_N$.

Performance prediction

- Number nodes in W in the decreasing order of their degrees: $F_1 \geqslant F_2 \geqslant \cdots \geqslant F_N$.
- S_j is the number of followers of node $j = 1, 2, \ldots, N$ among the n_1 randomly chosen nodes in V
- $S_j \sim Binomial(n_1, F_j/N)$

Performance prediction

- Number nodes in W in the decreasing order of their degrees: $F_1 \geqslant F_2 \geqslant \cdots \geqslant F_N$.
- S_j is the number of followers of node $j = 1, 2, \ldots, N$ among the n_1 randomly chosen nodes in V
- $S_j \sim Binomial(n_1, F_j/N)$
- $S_{i_1} \geqslant S_{i_2} \geqslant \ldots \geqslant S_{i_N}$ be the order statistics of $S_1, \ldots, S_N$.
- Performance measure:

$$E[\text{fraction of correctly identified top-}k\text{ entities}]$$

$$= \frac{1}{k} \sum_{j=1}^{k} P(j \in \{i_1, \ldots, i_{n_2}\}). \quad (2)$$

Performance prediction

- Number nodes in W in the decreasing order of their degrees: $F_1 \geqslant F_2 \geqslant \cdots \geqslant F_N$.
- S_j is the number of followers of node $j = 1, 2, \ldots, N$ among the n_1 randomly chosen nodes in V
- $S_j \sim Binomial(n_1, F_j/N)$
- $S_{i_1} \geqslant S_{i_2} \geqslant \ldots \geqslant S_{i_N}$ be the order statistics of $S_1, \ldots, S_N$.
- Performance measure:

$$E[\text{fraction of correctly identified top-}k\text{ entities}]$$

$$= \frac{1}{k} \sum_{j=1}^{k} P(j \in \{i_1, \ldots, i_{n_2}\}). \qquad (2)$$

- Computation of $P(j \in \{i_1, \ldots, i_{n_2}\})$ is not feasible even if degrees are known

Poisson prediction

- $P(j \in \{i_1, \ldots, i_{n_2}\})$
  $= P(S_j > S_{i_{n_2}}) + P(S_j = S_{i_{n_2}}, j \in \{i_1, \ldots, i_{n_2}\})$
- **Example.** Twitter graph, take $n_1 = n_2 = 500$. Then the average number of nodes i with $S_i = 1$ among the top-l nodes is

$$\sum_{i=1}^{l} P(S_i = 1) = \sum_{i=1}^{l} 500 \, \frac{F_i}{5 \cdot 10^8} \left(1 - \frac{F_i}{5 \cdot 10^8}\right)^{499},$$

  which is 2540.6 for $l = 10,000$ and it is 57.4 for $l = n_2 = 500$. Hence, typically, $[S_{i_{500}} = 1]$. The event $[i \in \{i_1, \ldots, i_{n_2}\}]$ occurs only for a small fraction of nodes i with $[S_i = 1]$.

Poisson prediction

- $P(j \in \{i_1, \ldots, i_{n_2}\})$
  $= P(S_j > S_{i_{n_2}}) + P(S_j = S_{i_{n_2}}, j \in \{i_1, \ldots, i_{n_2}\})$
- **Example.** Twitter graph, take $n_1 = n_2 = 500$. Then the average number of nodes i with $S_i = 1$ among the top-l nodes is

$$\sum_{i=1}^{l} P(S_i = 1) = \sum_{i=1}^{l} 500 \, \frac{F_i}{5 \cdot 10^8} \left(1 - \frac{F_i}{5 \cdot 10^8}\right)^{499},$$

  which is 2540.6 for $l = 10,000$ and it is 57.4 for $l = n_2 = 500$. Hence, typically, $[S_{i_{500}} = 1]$. The event $[i \in \{i_1, \ldots, i_{n_2}\}]$ occurs only for a small fraction of nodes i with $[S_i = 1]$.
- Approximation:
  $P(j \in \{i_1, \ldots, i_{n_2}\}) \approx P(S_j > S_{i_{n_2}}) \approx P(S_j > S_{n_2})$

Poisson prediction

- $P(j \in \{i_1, \ldots, i_{n_2}\})$
  $= P(S_j > S_{i_{n_2}}) + P(S_j = S_{i_{n_2}}, j \in \{i_1, \ldots, i_{n_2}\})$
- **Example.** Twitter graph, take $n_1 = n_2 = 500$. Then the average number of nodes i with $S_i = 1$ among the top-l nodes is

$$\sum_{i=1}^{l} P(S_i = 1) = \sum_{i=1}^{l} 500 \, \frac{F_i}{5 \cdot 10^8} \left(1 - \frac{F_i}{5 \cdot 10^8}\right)^{499},$$

  which is 2540.6 for $l = 10,000$ and it is 57.4 for $l = n_2 = 500$. Hence, typically, $[S_{i_{500}} = 1]$. The event $[i \in \{i_1, \ldots, i_{n_2}\}]$ occurs only for a small fraction of nodes i with $[S_i = 1]$.
- Approximation:
  $P(j \in \{i_1, \ldots, i_{n_2}\}) \approx P(S_j > S_{i_{n_2}}) \approx P(S_j > S_{n_2})$
- Assume F_j and F_{n_2} are known, then approximate $S_j \sim Poisson(n_1 F_j / N)$

EVT predictions

- Poisson approximation is not realistic: degrees are unknown

EVT predictions

- Poisson approximation is not realistic: degrees are unknown
- The algorithm finds a few highest degrees with accuracy almost 100%
- Let $\hat{F}_1 \geqslant \hat{F}_2 \geqslant \cdots \geqslant \hat{F}_m$ be the top-m degrees found by the algorithm, $m < k$

EVT predictions

- Poisson approximation is not realistic: degrees are unknown
- The algorithm finds a few highest degrees with accuracy almost 100%
- Let $\hat{F}_1 \geqslant \hat{F}_2 \geqslant \cdots \geqslant \hat{F}_m$ be the top-m degrees found by the algorithm, $m < k$
- The degrees follow a power law distribution with exponent γ

EVT predictions

- Poisson approximation is not realistic: degrees are unknown
- The algorithm finds a few highest degrees with accuracy almost 100%
- Let $\hat{F}_1 \geqslant \hat{F}_2 \geqslant \cdots \geqslant \hat{F}_m$ be the top-m degrees found by the algorithm, $m < k$
- The degrees follow a power law distribution with exponent γ
- Hill's estimator:

$$\hat{\gamma} = \left(\frac{1}{m-1} \sum_{i=1}^{m-1} \log(\hat{F}_i) - \log(\hat{F}_m) \right)^{-1}. \qquad (3)$$

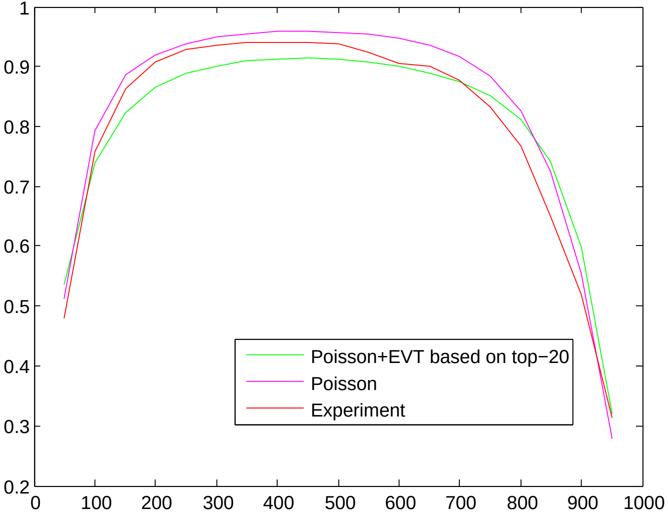
EVT predictions

- Poisson approximation is not realistic: degrees are unknown
- The algorithm finds a few highest degrees with accuracy almost 100%
- Let $\hat{F}_1 \geqslant \hat{F}_2 \geqslant \cdots \geqslant \hat{F}_m$ be the top-m degrees found by the algorithm, $m < k$
- The degrees follow a power law distribution with exponent γ
- Hill's estimator:

$$\hat{\gamma} = \left(\frac{1}{m-1} \sum_{i=1}^{m-1} \log(\hat{F}_i) - \log(\hat{F}_m) \right)^{-1}. \qquad (3)$$

- Estimator for high degrees: Dekkers et al. (1989)
  $\hat{f}_j = \hat{F}_m \left(\frac{m}{j-1} \right)^{1/\hat{\gamma}}, \qquad j > 1, j << N.$
- Use $S_j \sim Poisson(n_1 \hat{f}_j / N)$

Performance predictions on the Twitter graph



Sublinear complexity

- $1, \ldots, k$ – top-k nodes in W; $F_1, \ldots, F_k$ – their degrees

Sublinear complexity

- $1, \dots, k$ – top-k nodes in W; $F_1, \dots, F_k$ – their degrees
- $S_j \sim Binomial(n_1, F_j/N)$

Sublinear complexity

- $1, \ldots, k$ – top-k nodes in W; $F_1, \ldots, F_k$ – their degrees
- $S_j \sim Binomial(n_1, F_j/N)$
- With normal approximation, and error pr-ty α we need that

$$\sqrt{\frac{n_1}{N}} \frac{F_k - F_{n_2}}{\sqrt{F_k + F_{n_2}}} > z_{1-\alpha}$$

Sublinear complexity

- $1, \ldots, k$ – top-k nodes in W; $F_1, \ldots, F_k$ – their degrees
- $S_j \sim Binomial(n_1, F_j/N)$
- With normal approximation, and error pr-ty α we need that

$$\sqrt{\frac{n_1}{N}} \frac{F_k - F_{n_2}}{\sqrt{F_k + F_{n_2}}} > z_{1-\alpha}$$

- $F_k >> F_{n_2}$

Sublinear complexity

- $1, \dots, k$ – top-k nodes in W; $F_1, \dots, F_k$ – their degrees
- $S_j \sim Binomial(n_1, F_j/N)$
- With normal approximation, and error pr-ty α we need that

$$\sqrt{\frac{n_1}{N}} \frac{F_k - F_{n_2}}{\sqrt{F_k + F_{n_2}}} > z_{1-\alpha}$$

- $F_k >> F_{n_2}$
- Assuming the i.i.d. degrees, by the Extreme Value Theory, w.h.p., $\log(F_k) = \gamma^{-1} \log(N)(1 + o(log(N)))$

Sublinear complexity

- $1, \ldots, k$ – top-k nodes in W; $F_1, \ldots, F_k$ – their degrees
- $S_j \sim Binomial(n_1, F_j/N)$
- With normal approximation, and error pr-ty α we need that

$$\sqrt{\frac{n_1}{N}} \frac{F_k - F_{n_2}}{\sqrt{F_k + F_{n_2}}} > z_{1-\alpha}$$

- $F_k >> F_{n_2}$
- Assuming the i.i.d. degrees, by the Extreme Value Theory, w.h.p., $\log(F_k) = \gamma^{-1} \log(N)(1 + o(log(N)))$
- Roughly, $n_1 = O(N^{1-1/\gamma})$

Sublinear complexity

- $1, \ldots, k$ – top-k nodes in W; $F_1, \ldots, F_k$ – their degrees
- $S_j \sim Binomial(n_1, F_j/N)$
- With normal approximation, and error pr-ty α we need that

$$\sqrt{\frac{n_1}{N}} \frac{F_k - F_{n_2}}{\sqrt{F_k + F_{n_2}}} > z_{1-\alpha}$$

- $F_k >> F_{n_2}$
- Assuming the i.i.d. degrees, by the Extreme Value Theory, w.h.p., $\log(F_k) = \gamma^{-1} \log(N)(1 + o(log(N)))$
- Roughly, $n_1 = O(N^{1-1/\gamma})$
- Since $\sum_w S_w = O(n_1)$ w.h.p., n_2 is at most $O(n_1)$

Sublinear complexity

- $1, \ldots, k$ – top-k nodes in W; $F_1, \ldots, F_k$ – their degrees
- $S_j \sim Binomial(n_1, F_j/N)$
- With normal approximation, and error pr-ty α we need that

$$\sqrt{\frac{n_1}{N}} \frac{F_k - F_{n_2}}{\sqrt{F_k + F_{n_2}}} > z_{1-\alpha}$$

- $F_k >> F_{n_2}$
- Assuming the i.i.d. degrees, by the Extreme Value Theory, w.h.p., $\log(F_k) = \gamma^{-1} \log(N)(1 + o(log(N)))$
- Roughly, $n_1 = O(N^{1-1/\gamma})$
- Since $\sum_w S_w = O(n_1)$ w.h.p., n_2 is at most $O(n_1)$
- We conclude that roughly $n = n_1 + n_2 = O(N^{1-1/\gamma})$

Sublinear complexity

- $1, \ldots, k$ – top-k nodes in W; $F_1, \ldots, F_k$ – their degrees
- $S_j \sim Binomial(n_1, F_j/N)$
- With normal approximation, and error pr-ty α we need that

$$\sqrt{\frac{n_1}{N}} \frac{F_k - F_{n_2}}{\sqrt{F_k + F_{n_2}}} > z_{1-\alpha}$$

- $F_k >> F_{n_2}$
- Assuming the i.i.d. degrees, by the Extreme Value Theory, w.h.p., $\log(F_k) = \gamma^{-1} \log(N)(1 + o(log(N)))$
- Roughly, $n_1 = O(N^{1-1/\gamma})$
- Since $\sum_w S_w = O(n_1)$ w.h.p., n_2 is at most $O(n_1)$
- We conclude that roughly $n = n_1 + n_2 = O(N^{1-1/\gamma})$
- Note that the complexity is in terms of $|W| = N$

Sublinear complexity

- $1, \ldots, k$ – top-k nodes in W; $F_1, \ldots, F_k$ – their degrees
- $S_j \sim Binomial(n_1, F_j/N)$
- With normal approximation, and error pr-ty α we need that

$$\sqrt{\frac{n_1}{N}} \frac{F_k - F_{n_2}}{\sqrt{F_k + F_{n_2}}} > z_{1-\alpha}$$

- $F_k >> F_{n_2}$
- Assuming the i.i.d. degrees, by the Extreme Value Theory, w.h.p., $\log(F_k) = \gamma^{-1} \log(N)(1 + o(log(N)))$
- Roughly, $n_1 = O(N^{1-1/\gamma})$
- Since $\sum_w S_w = O(n_1)$ w.h.p., n_2 is at most $O(n_1)$
- We conclude that roughly $n = n_1 + n_2 = O(N^{1-1/\gamma})$
- Note that the complexity is in terms of $|W| = N$
- High variability helps a lot!

Thank you!