http://poloclub.gatech.edu/cse6242

CSE6242 / CX4242: Data & Visual Analytics

Graphs / Networks

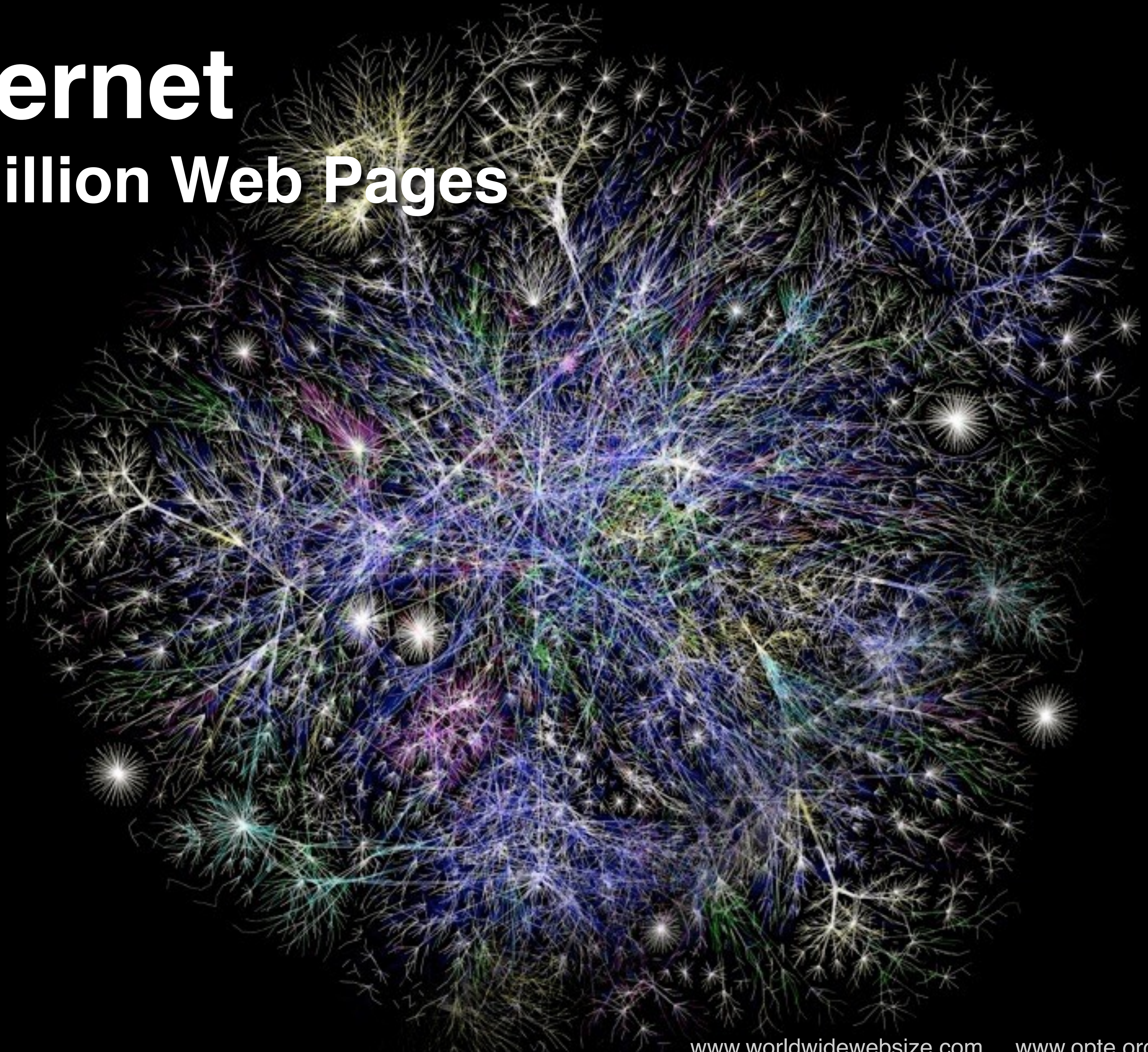
Basics, how to build & store graphs, laws, etc.
Centrality, and algorithms you should know

Duen Horng (Polo) Chau
Associate Professor
Associate Director, MS Analytics
Machine Learning Area Leader, College of Computing
Georgia Tech

Partly based on materials by
Professors Guy Lebanon, Jeffrey Heer, John Stasko, Christos Faloutsos, Parishit Ram (GT PhD alum; IBM), Alex Gray

Internet

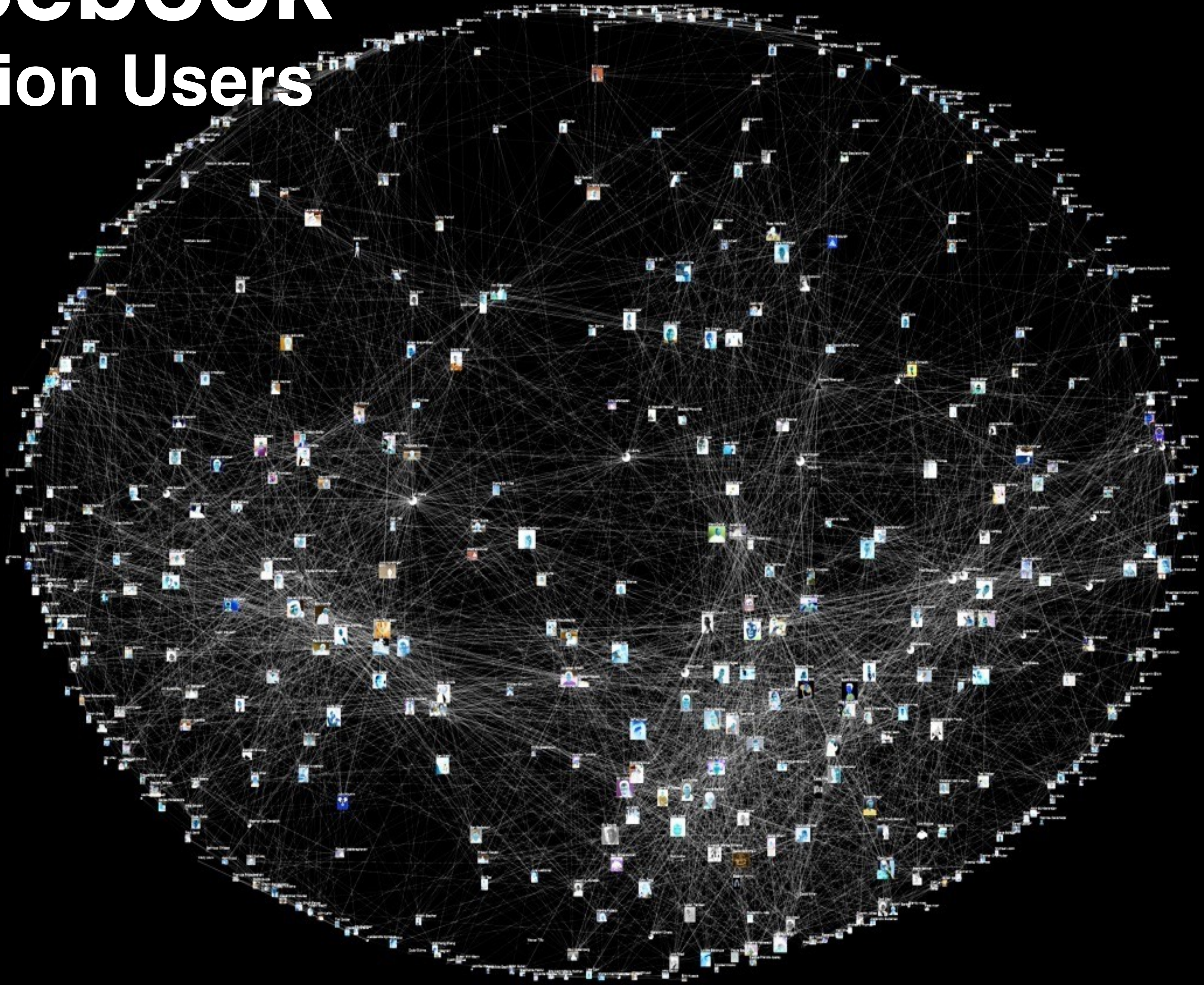
4 Billion Web Pages



www.worldwidewebsize.com www.opte.org

Facebook

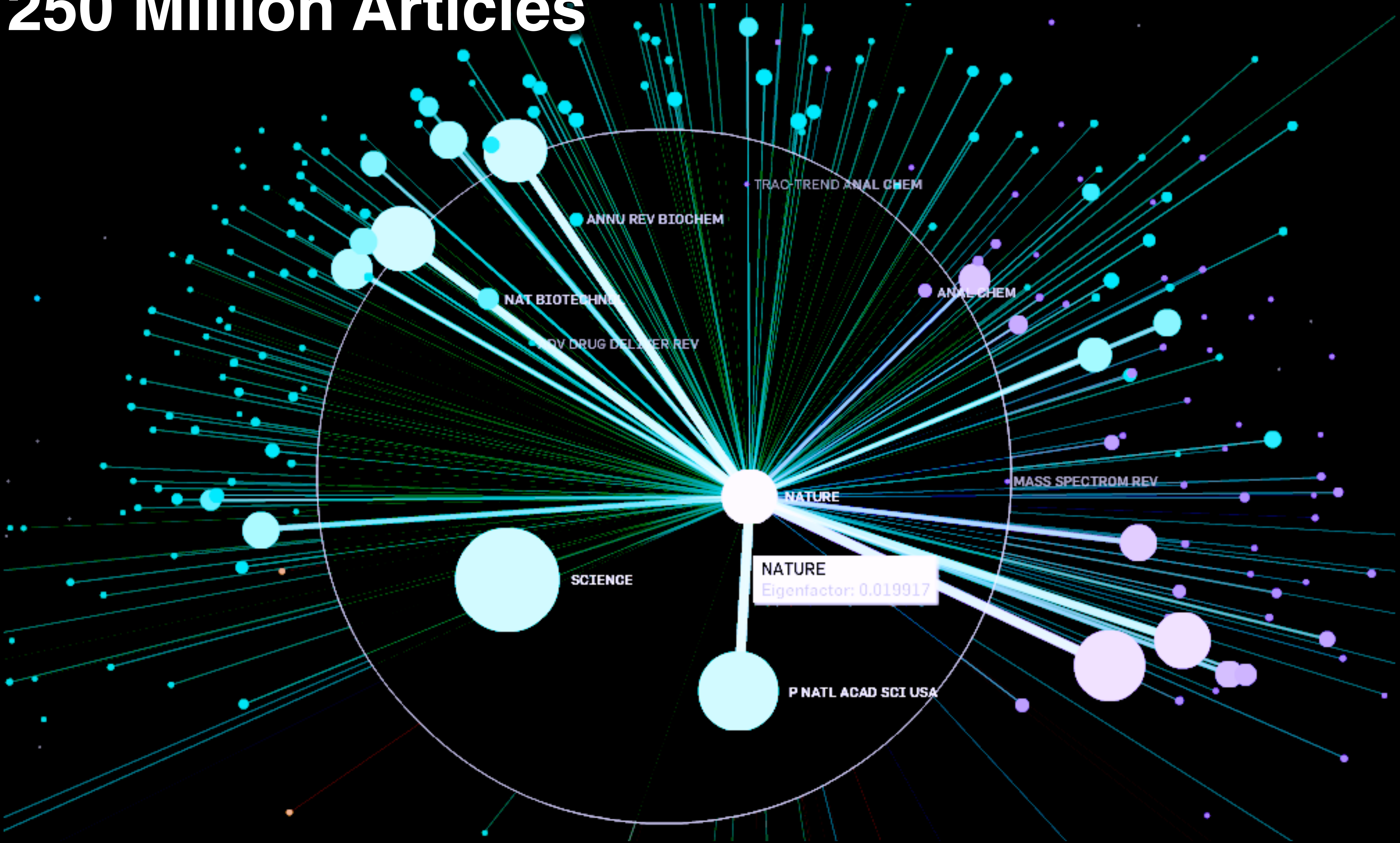
2 Billion Users



Modified from Marc_Smith, flickr

Citation Network

250 Million Articles



www.scirus.com/press/html/feb_2006.html#2 Modified from well-formed.eigenfactor.org

Many More

twitter

Who-follows-whom (288 million users)

amazon

Who-buys-what (120 million users)

at&t cellphone network

Who-calls-whom (100 million users)

Protein-protein interactions

200 million possible interactions in human genome

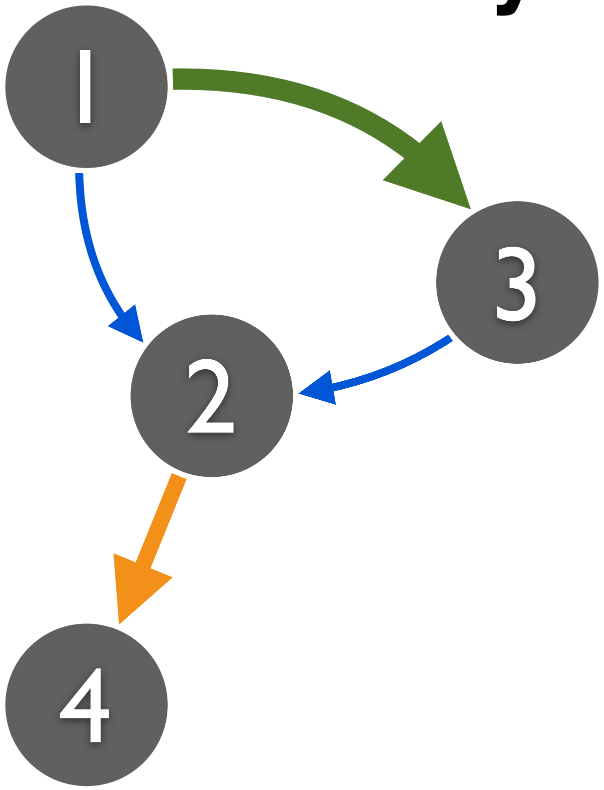
Sources: www.selectscience.net www.phonedog.com www.mediabistro.com www.practicalecommerce.com/

How to **represent** a graph?

Conceptually.
Visually.
Programmatically.

How to **Represent** a Graph?

Visually



Adjacency matrix

Target node

Source node

	1	2	3	4
1	0	1	3	0
2	0	0	0	2
3	0	1	0	0
4	0	0	0	0

Adjacency list

1: 2, 3
2: 4
3: 2

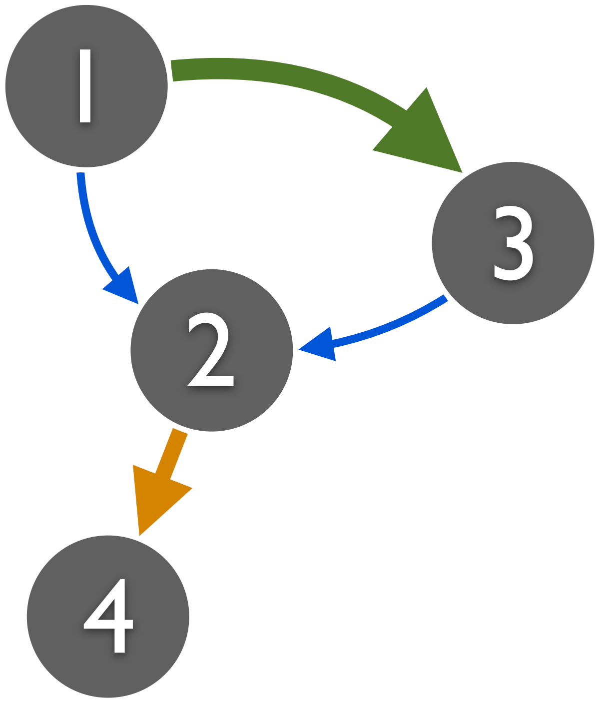
Edge list

1, 2, 1
1, 3, 3
2, 4, 2
3, 2, 1

- most common distribution format
- sometimes **painful** to parse when edges/nodes have many columns (some are text with double/single quotes, some are integers, some decimals, ...)

How to Represent a Graph?

Visually



Adjacency matrix

Target node

Source node

	1	2	3	4
1	0	1	3	0
2	0	0	0	2
3	0	1	0	0
4	0	0	0	0

Adjacency list

1: 2, 3
2: 4
3: 2

Edge list

1, 2, 1
1, 3, 3
2, 4, 2
3, 2, 1

Each node is often identified by a numeric ID. Why?

Assigning an ID to a node

- Use a “map” (Java) / “dictionary” (Python) / SQLite
- Same concept: given an entity/node (e.g., “Tom”) not seen before, assign a number to it
- Example of using SQLite to map names to IDs

Hidden column; SQLite automatically created for you

rowid	**name**
1	Tom
2	Sandy
3	Richard
4	Polo

How to use the node IDs?

Create an index for “name”. Then write a “join” query.

rowid	**name**
1	Tom
2	Sandy
3	Richard
4	Polo

source	**target**
Tom	Sandy
Polo	Richard

source	**target**
1	2
4	3

How to store “large” graphs?

How large is “large”?

What do you think?

- In what units? Thousands? Millions?

How do you measure a graph’s size?

- By ...

(Hint: highly subjective. And domain specific.)

Storing large graphs...

On your laptop computer

- SQLite
- Neo4j (GPL license)
  http://neo4j.com/licensing/

On a server

- MySQL, PostgreSQL, etc.
- Neo4j (?)

Storing large graphs...

With a cluster

- **Titan** (on top of **HBase**), S2Graph — if you need real time read and write
- **Hadoop** (generic framework) — if batch processing is fine
- **Hama, Giraph**, inspired by Google's Pregel
- **FlockDB**, by Twitter
- Turri (Apple) / Dato / GraphLab

Storing large graphs on your computer

I like to use **SQLite**. Why? **Good enough for my use.**

- Easily handle up to **gigabytes**
  - Roughly **tens of millions** of nodes/edges (perhaps up to billions?). Very good! For **today's** standard.
- Very easy to maintain: **one** cross-platform file
- Has programming wrappers in numerous languages
  - C++, Java (Andriod), Python, Objective C (iOS),...
- Queries are so easy!
  e.g., find all nodes' degrees = 1 SQL statement
- Bonus: SQLite even supports **full-text search**
- Offline application support (iPad)

SQLite graph database schema

Simplest schema:

```
edges(source_id, target_id)
```

More sophisticated (flexible; lets you store more things):

```
CREATE TABLE nodes (
   id INTEGER PRIMARY KEY,
   type INTEGER DEFAULT 0,
   name VARCHAR DEFAULT '');

CREATE TABLE edges (
   source_id INTEGER,
   target_id INTEGER,
   type INTEGER DEFAULT 0,
   weight FLOAT DEFAULT 1,
   timestamp INTEGER DEFAULT 0,
   PRIMARY KEY(source_id, target_id, timestamp));
```

[Side note; you already did this in HW1]

Full-Text Search (FTS) on SQLite

http://www.sqlite.org/fts3.html

Very simple. Built-in. Only needs 3 lines of commands.

- **Create** FTS table (index)

```
CREATE VIRTUAL TABLE critics_consensus USING
fts4(consensus);
```

- **Insert** text into FTS table

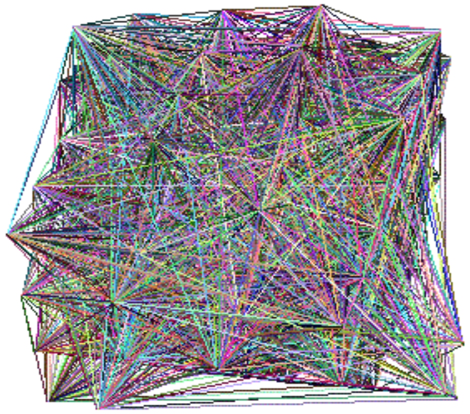
```
INSERT INTO critics_consensus SELECT
critics_consensus FROM movies;
```

- **Query** using the “match” keyword

```
SELECT * FROM critics_consensus WHERE consensus
MATCH 'funny OR horror';
```

SQLite originally developed by Google engineers

I have a graph dataset. Now what?



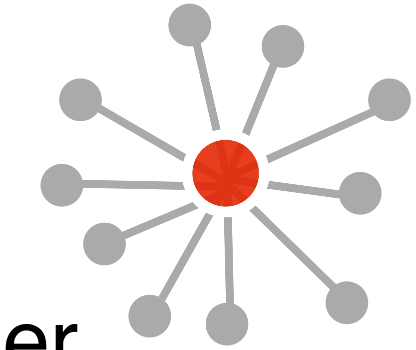
Yuck!

Analyze it! Do “**data mining**” or “**graph mining**”.

How does it “look like”? Visualize it if it’s small.

Does it follow any expected patterns?
Or does it *not* follow some expected patterns (outliers)?

- Why does this matter?
- If we know the **patterns** (models), we can do **prediction**, **recommendation**, etc.
  e.g., is Alice going to “friend” Bob on Facebook?
  People often buy beer and diapers together.
- **Outliers** often give us **new insights**
  e.g., telemarketer’s “friends” don’t know each other



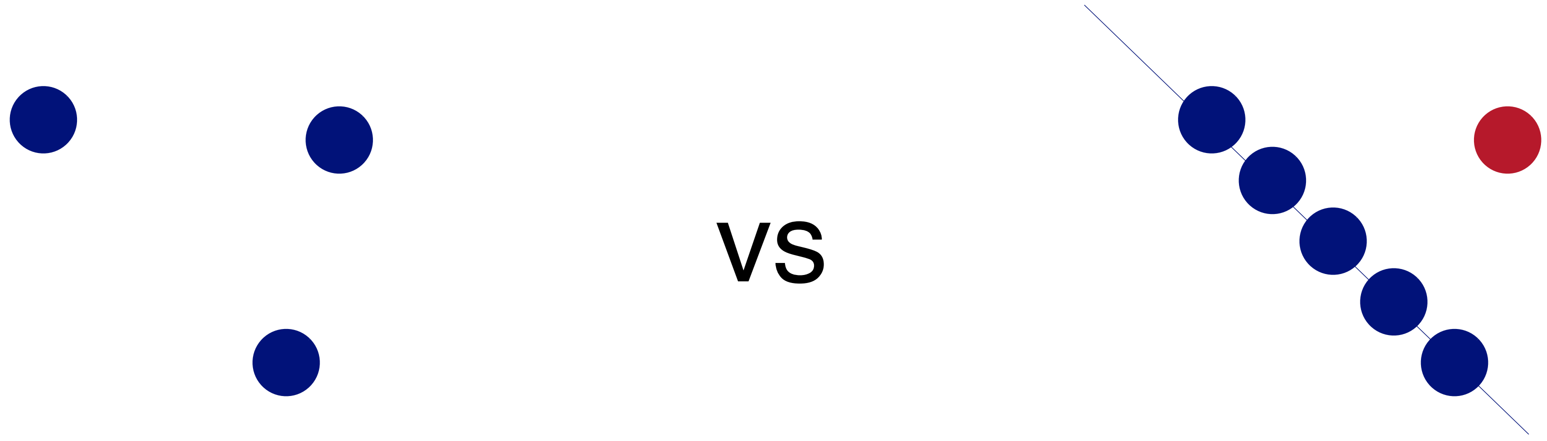
Finding patterns & outliers in graphs

Outlier/Anomaly detection

- To spot them, **we need to find patterns first**
- Anomalies = things that do not fit the patterns

To effectively do this, we need large datasets

- patterns and anomalies don’t show up well in small datasets



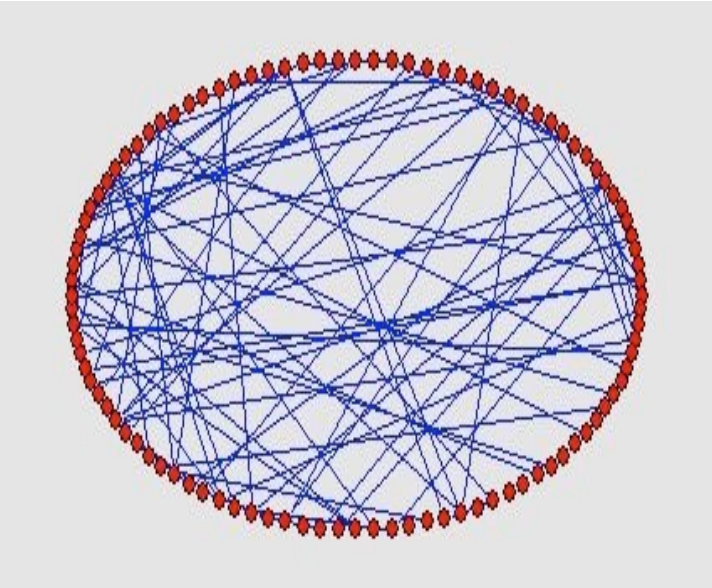
Are real graphs random?

Random graph (Erdos-Renyi)
100 nodes, avg degree = 2

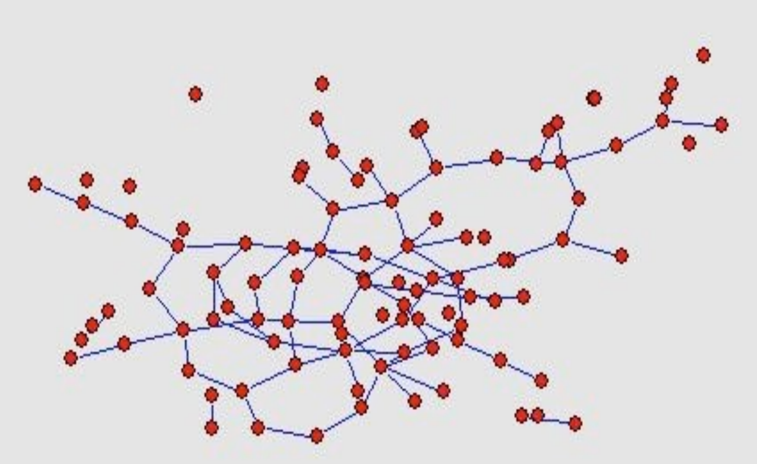
http://en.wikipedia.org/wiki/Erdős–Rényi_model

No obvious patterns

Before layout



After layout



Graph and layout generated with pajek

http://vlado.fmf.uni-lj.si/pub/networks/pajek/

Laws and patterns

Laws and patterns

- Are real graphs random?

Laws and patterns

- Are real graphs random?

Laws and patterns

- Are real graphs random?

- A: NO!!!
  - Diameter (longest shortest path)
  - in- and out- degree distributions
  - other (surprising) patterns
  - **So, let’s look at the data**

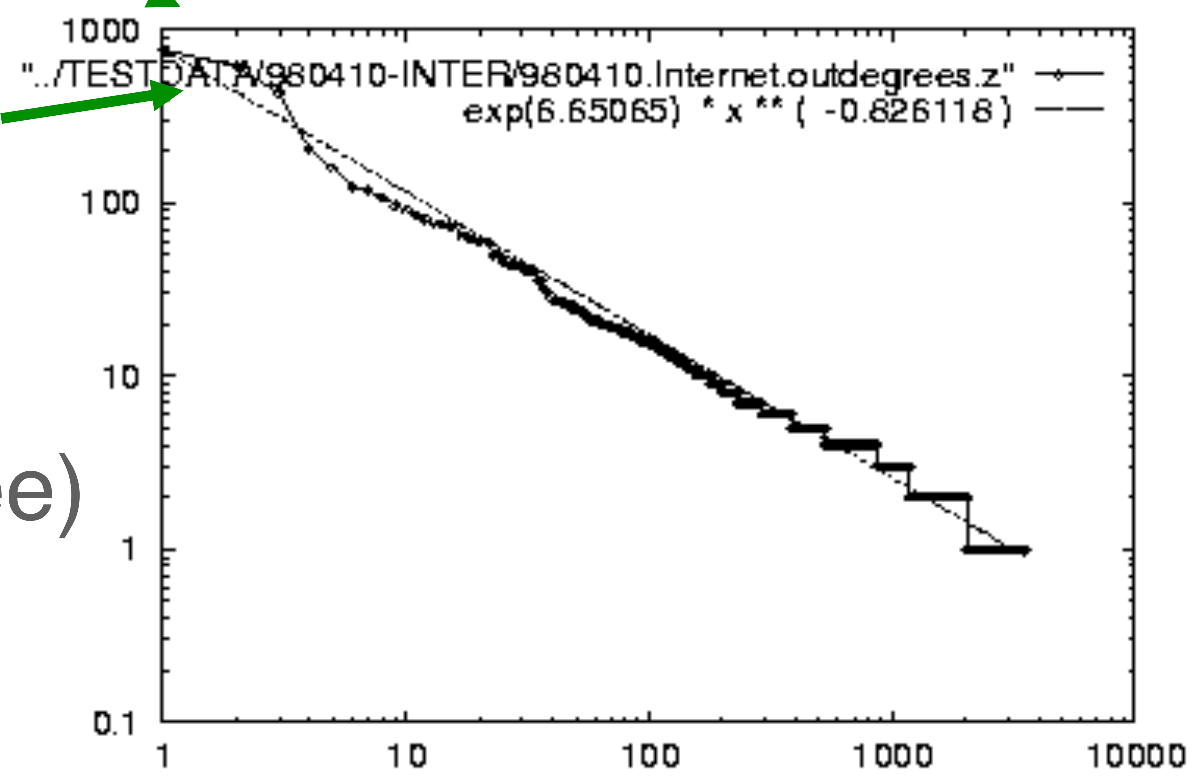
Power Law in Degree Distribution

Faloutsos, Faloutsos, Faloutsos [SIGCOMM99]
Seminal paper. Must read!

internet domains

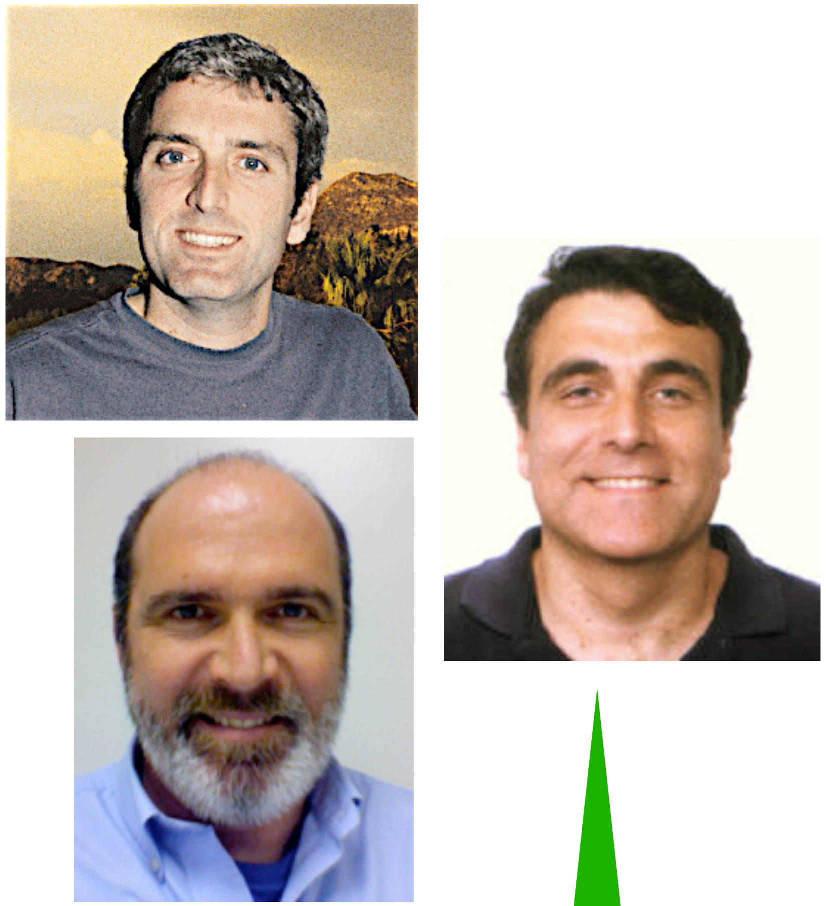
att.com

ibm.com



log(degree)

log(rank)



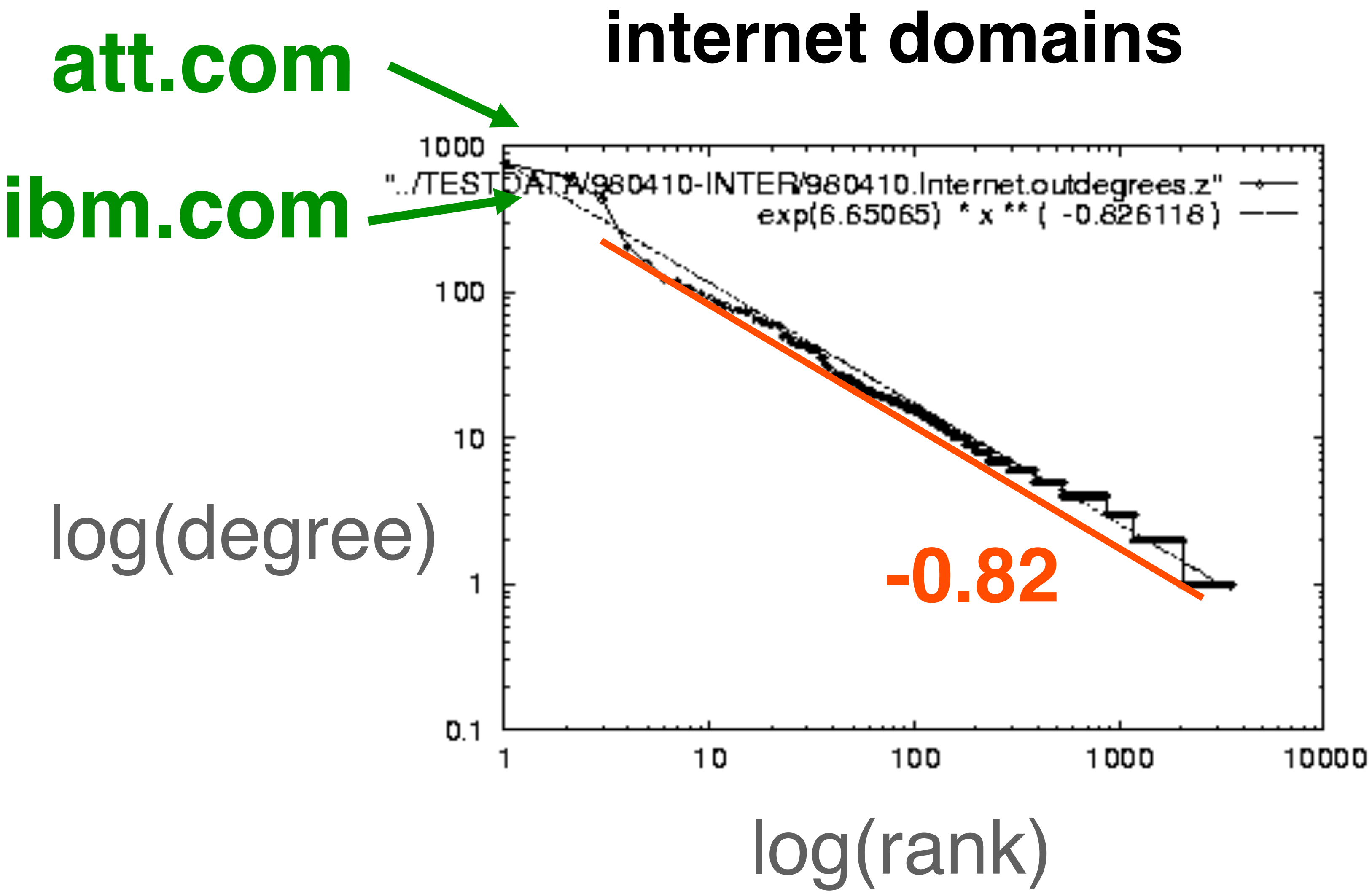
Christos was Polo's advisor

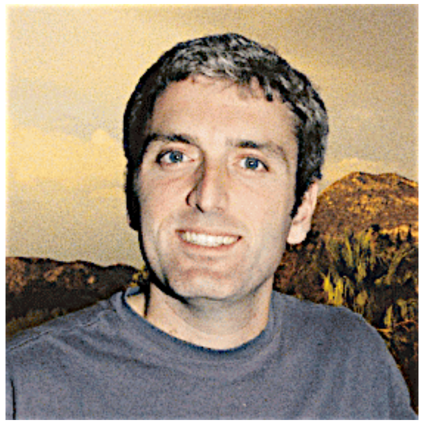
Zipf's law: the frequency of any item is **inversely proportional** to the item's rank (when ranked by decreasing frequency)

http://en.wikipedia.org/wiki/Zipf%27s_law

Power Law in Degree Distribution

Faloutsos, Faloutsos, Faloutsos [SIGCOMM99]
Seminal paper. Must read!



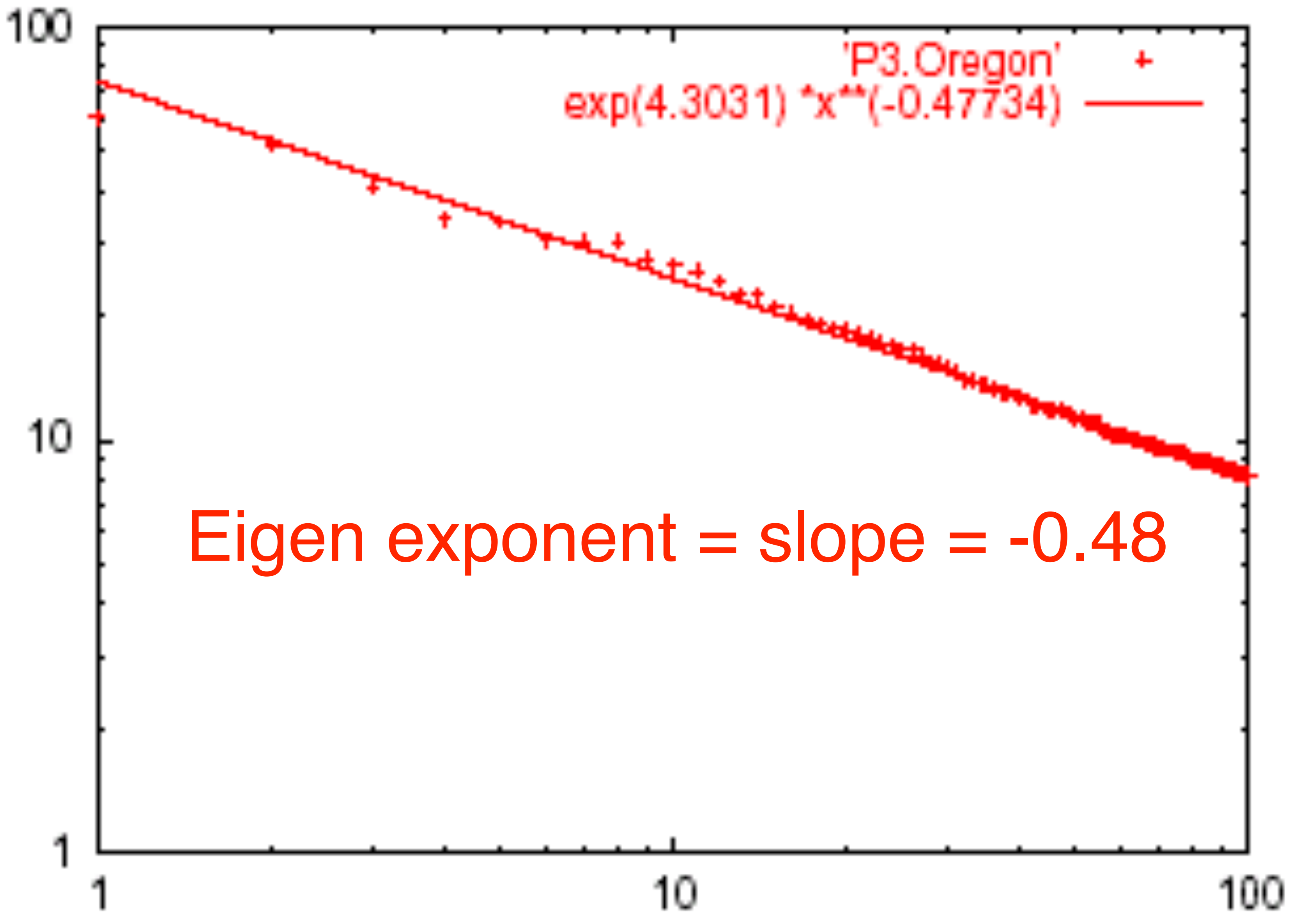


Christos was Polo’s advisor

Zipf’s law: the frequency of any item is **inversely proportional** to the item’s rank (when ranked by decreasing frequency)

http://en.wikipedia.org/wiki/Zipf%27s_law

Power Law in Eigenvalues of Adjacency Matrix



Eigenvalue

Eigen exponent = slope = -0.48

Rank of decreasing eigenvalue

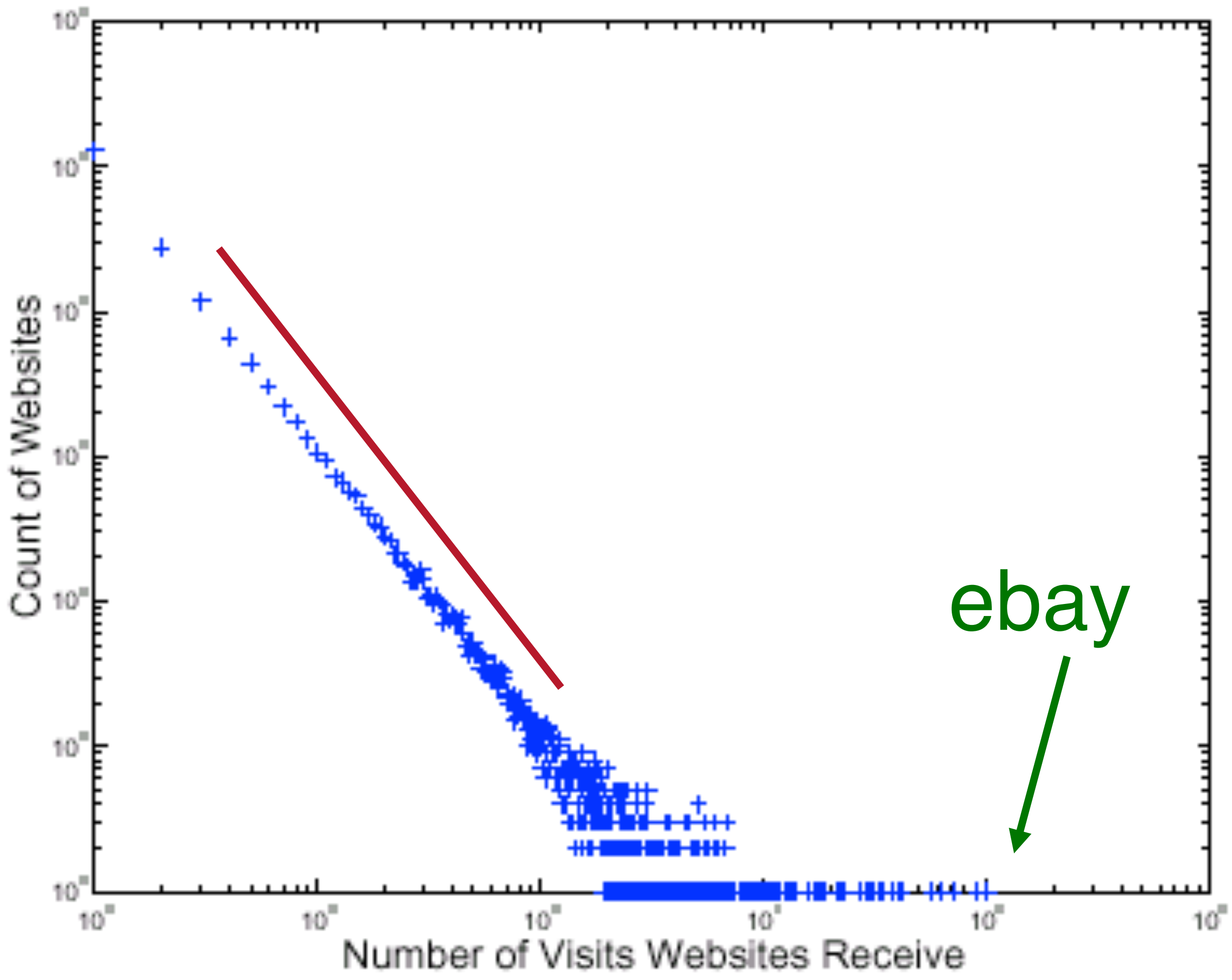
How about graphs from other domains?

More Power Laws

- Web hit counts
  [Alan L. Montgomery and Christos Faloutsos]

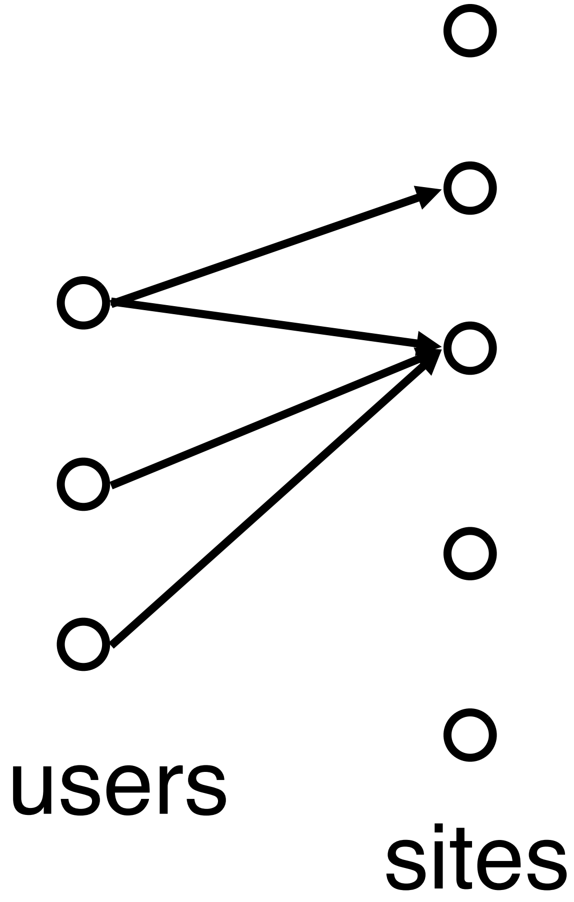
Web Site Traffic

log(#website)



ebay

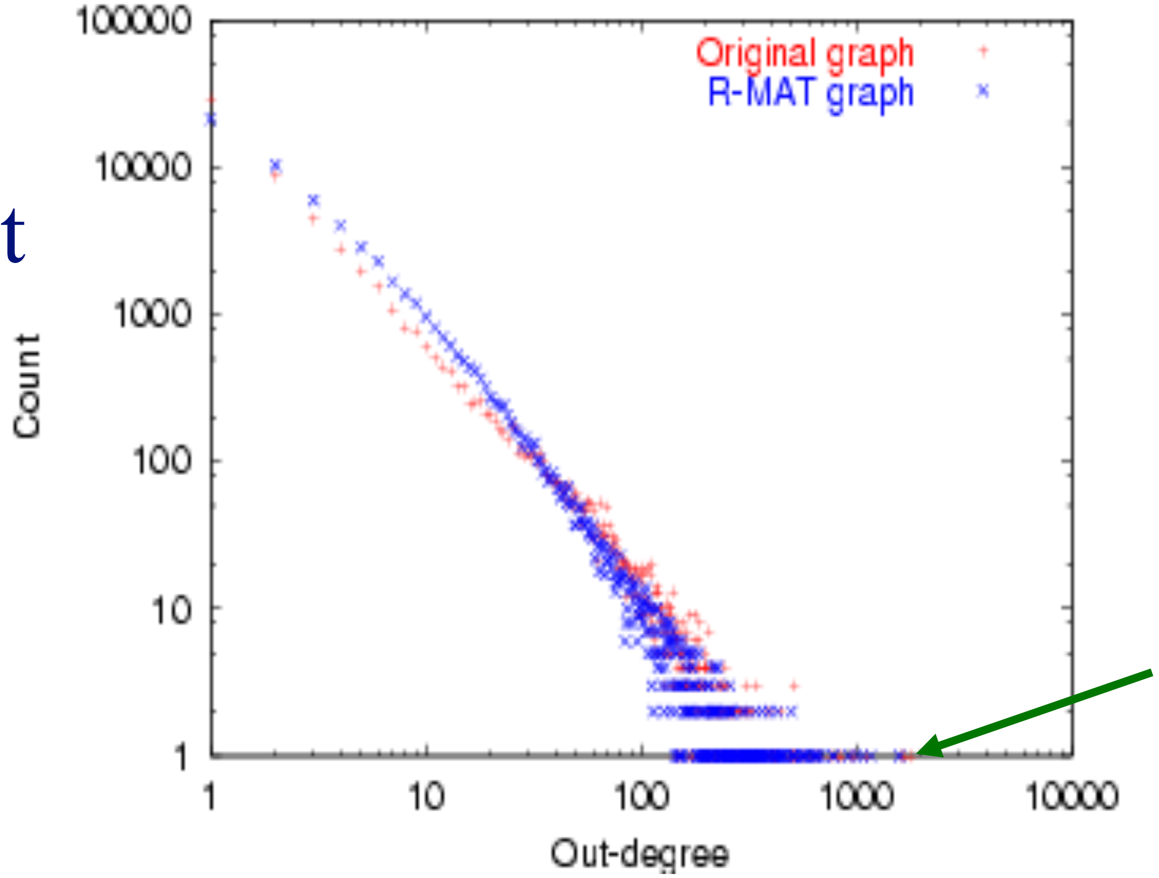
log(#website visit)



epinions.com

- who-trusts-whom [Richardson + Domingos, KDD 2001]

count



trusts-2000-people user

(out) degree

And numerous more

- # of sexual contacts
- Income [Pareto] – 80-20 distribution
- Duration of downloads [Bestavros+]
- Duration of UNIX jobs
- File sizes
- …

Any other ‘laws’?

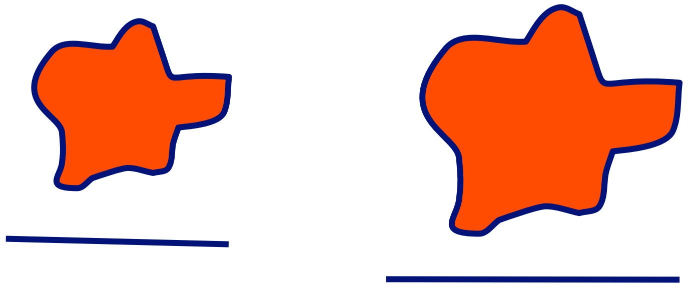
- Yes!
- Small diameter (~ constant!) –
  - six degrees of separation / ‘Kevin Bacon’
  - small worlds [Watts and Strogatz]

Problem: Time evolution

- Jure Leskovec (CMU -> Stanford)
- Jon Kleinberg (Cornell)
- Christos Faloutsos (CMU)

Evolution of the Diameter

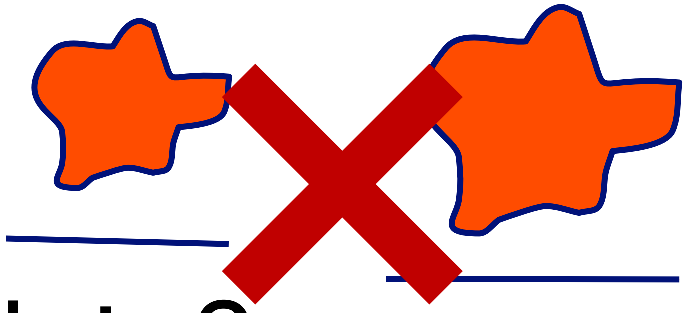
- Prior work on Power Law graphs hints at slowly growing diameter:
  - diameter ~ O(log N)
  - diameter ~ O(log log N)



- What is happening in real data?

Evolution of the Diameter

- Prior work on Power Law graphs hints at slowly growing diameter:
  - diameter ~ O(log N)
  - diameter ~ O(log log N)

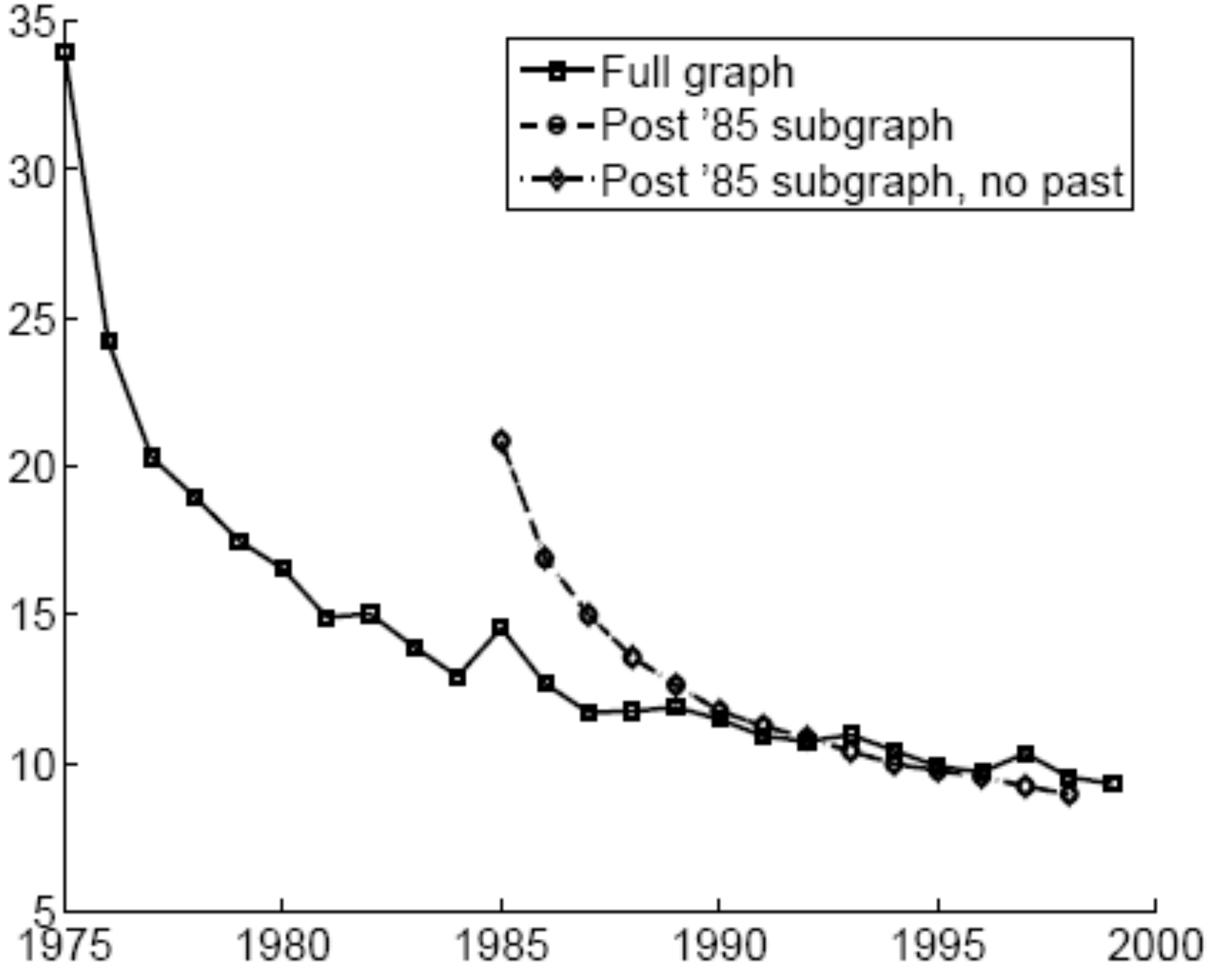


- What is happening in real data?
- Diameter shrinks over time

Diameter – Patents Network

- Patent citation network
- 25 years of data
- @1999
  - 2.9 M nodes
  - 16.5 M edges

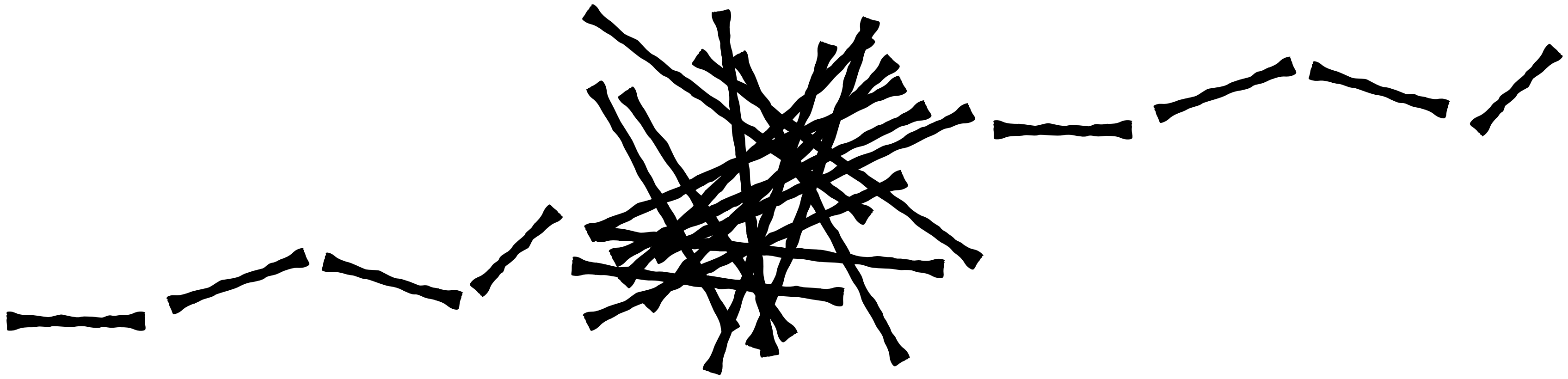
Effective diameter



Time (year)

Why **Effective Diameter**?

The **maximum** diameter is susceptible to **outliers**



So, we use **effective** diameter instead

- defined as the **minimum** number of hops in which **90% of connected node pairs** can reach each other

Evolution of #Node and #Edge

N(t) … nodes at time t

E(t) … edges at time t

Suppose that

$$N(t+1) = 2 * N(t)$$

Q: what is your guess for

$$E(t+1) =? 2 * E(t)$$

Evolution of #Node and #Edge

N(t) … nodes at time t

E(t) … edges at time t

Suppose that

$$N(t+1) = 2 * N(t)$$

Q: what is your guess for

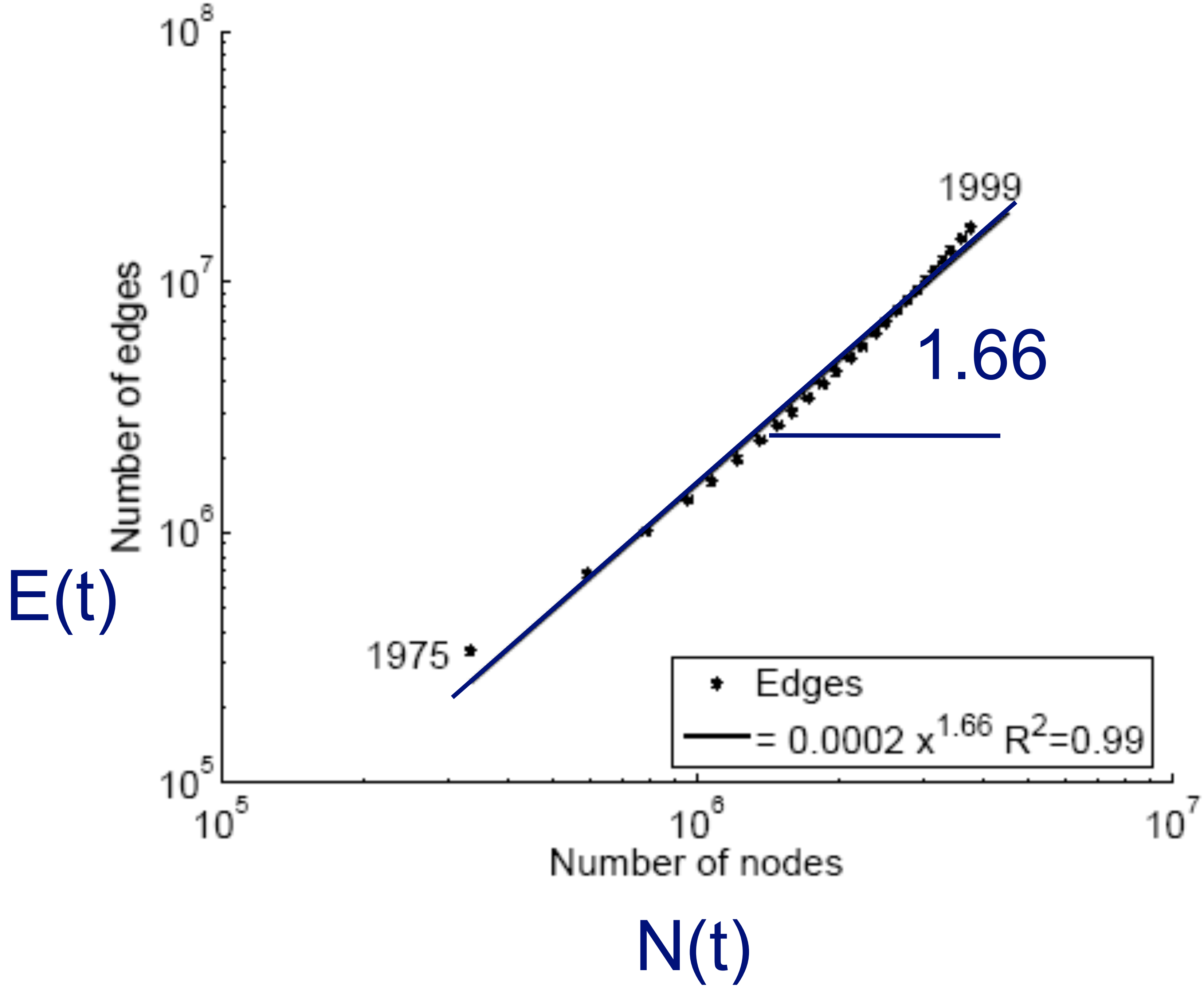
$$E(t+1) =? 2 * E(t)$$ 🚫

A: over-doubled!

But obeying the “Densification Power Law” 😯

Densification – Patent Citations

- Citations among patents granted
- @1999
  - 2.9 M nodes
  - 16.5 M edges
- Each year is a datapoint

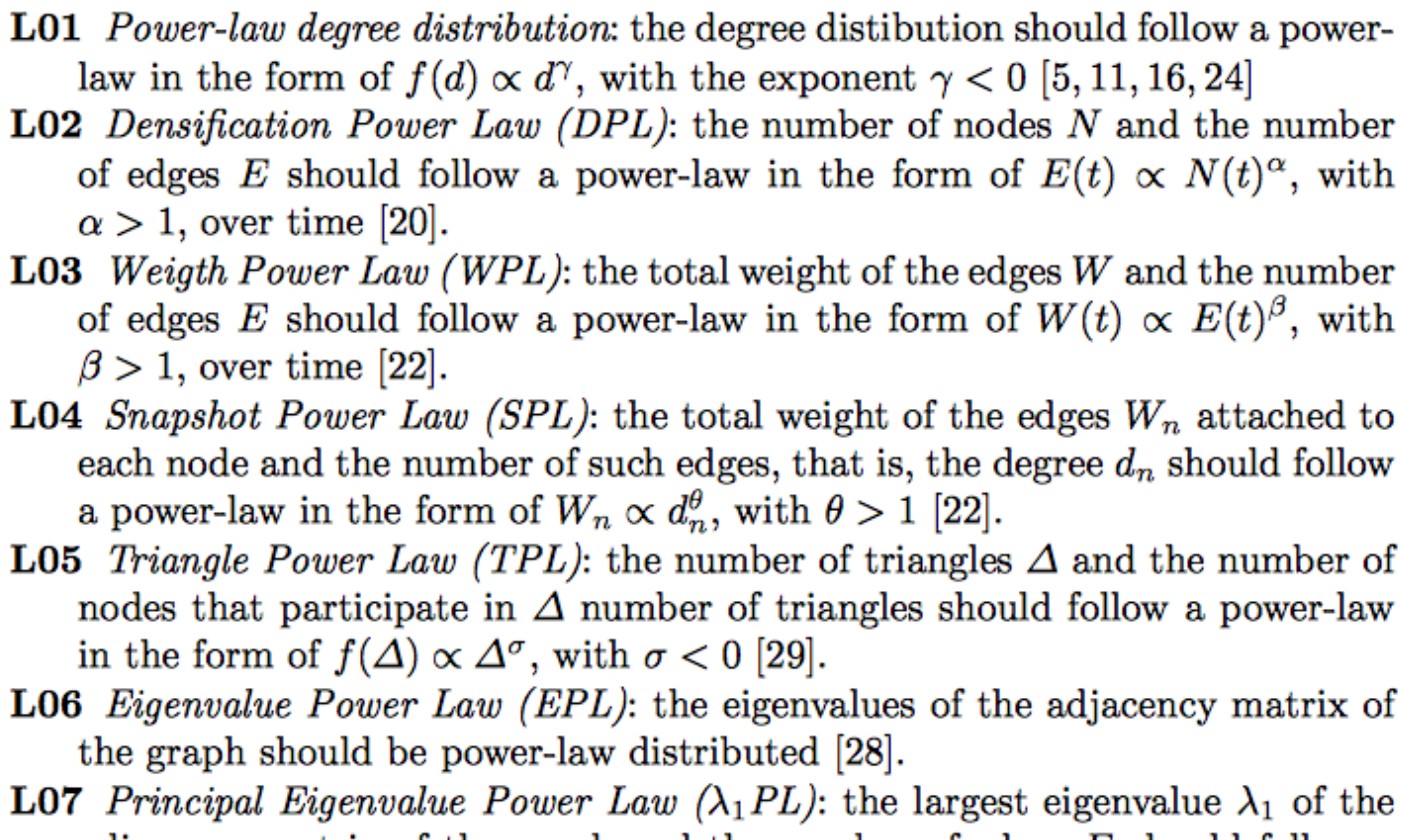


So many laws!

There will be more to come...

To date, there are **11 (or more) laws**

- RTG: A Recursive Realistic Graph Generator using Random Typing [Akoglu, Faloutsos]



L01 *Power-law degree distribution*: the degree distibution should follow a power-law in the form of $f(d) \propto d^{\gamma}$, with the exponent $\gamma < 0$ [5, 11, 16, 24]

L02 *Densification Power Law (DPL)*: the number of nodes N and the number of edges E should follow a power-law in the form of $E(t) \propto N(t)^{\alpha}$, with $\alpha > 1$, over time [20].

L03 *Weigth Power Law (WPL)*: the total weight of the edges W and the number of edges E should follow a power-law in the form of $W(t) \propto E(t)^{\beta}$, with $\beta > 1$, over time [22].

L04 *Snapshot Power Law (SPL)*: the total weight of the edges W_n attached to each node and the number of such edges, that is, the degree d_n should follow a power-law in the form of $W_n \propto d_n^{\theta}$, with $\theta > 1$ [22].

L05 *Triangle Power Law (TPL)*: the number of triangles Δ and the number of nodes that participate in Δ number of triangles should follow a power-law in the form of $f(\Delta) \propto \Delta^{\sigma}$, with $\sigma < 0$ [29].

L06 *Eigenvalue Power Law (EPL)*: the eigenvalues of the adjacency matrix of the graph should be power-law distributed [28].

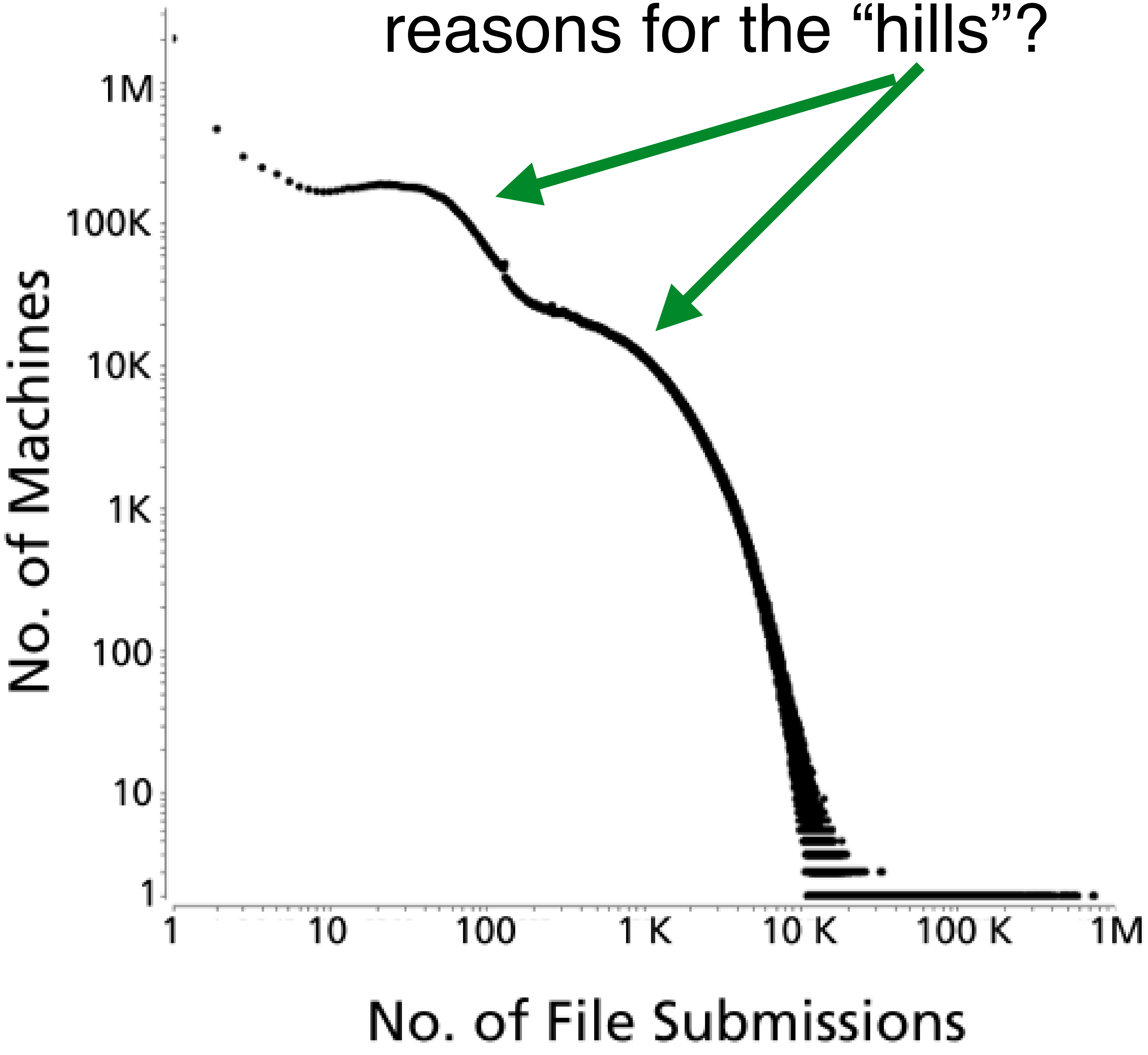
L07 *Principal Eigenvalue Power Law ($\lambda_1 PL$)*: the largest eigenvalue λ_1 of the

So many laws!

What should you do?

- **Try as many distributions as possible** and see if your graph fits them.
- **If it doesn't, find out the reasons.** Sometimes it's due to errors/problems in the data; sometimes, it signifies some new patterns!

What might be the reasons for the “hills”?



Polonium: Tera-Scale Graph Mining and Inference for Malware Detection [Chau, et al]