Case Study 1 : Graph Structure & Modelling

Broder et al, Power-Law & Price's Model

Author :	*Paula Dwan*
Email :	*paula.dwan@gmail.com*
Student ID :	*13208660*
Course :	*MSc Advanced Software Engineering*
Module :	*COMP-47270 Computational Network Analysis and Modelling*
Lecturer :	*Dr. Neil Hurley*
Email :	*neil.hurley@ucd.ie*
Due Date :	*11 May 2015*

Table of Contents

1 **Case Study Requirements** 3

1.1 Understanding of Requirements from Class Discussions 3

2 **Introduction** 4

2.1 What is a Directed Network? 4

2.2 Directed Graphs Used 4

3 **Broder et al.** 4

3.1 Overview of Broder et al. 4

3.1.1 Components : Bow-Tie Model 5

3.1.2 How Calculated Using Python 5

3.2 Results of Broder et al. 7

3.2.1 Small graph → ≤ 10k nodes 7

3.2.2 Medium graph→ 10k to 100k nodes 7

3.2.3 Large graph → > 100k nodes 7

4 **Plot the Degree Distribution** 8

4.1 Overview 8

4.1.1 Components : Log-Log Plot 8

4.1.2 How Calculated Using Python 8

4.2 Results 9

4.2.1 Small graph → ≤ 10k nodes : Power Law Distribution 9

4.2.2 Medium graph→ 10k to 100k nodes : Power Law Distribution 9

4.2.3 Large graph → > 100k nodes : Power Law Distribution 10

4.2.4 Parameter β for Power-Law Degree Distribution Model 10

5 **Price's Model** 10

5.1 Overview 10

5.1.1 Components : Price's Model 10

5.1.2 How Calculated Using Python 11

5.2 Results 11

5.2.1 Small graph → ≤ 10k nodes : Price's Model 11

5.2.2 Medium graph→ 10k to 100k nodes : Price's Model 11

5.2.3 Large graph → > 100k nodes : Price's Model 11

5.2.4 Alpha (a) : Price's Model 12

5.2.5 Networks best represented by Price's Model 12

6 **Conclusions** 12

7 **References – Datasets** 13

7.1 Networks Datasets 13

7.1.1 Dataset Statistics – Datasets 13

7.1.2 Small Dataset information – http://snap.stanford.edu/data/egonets-Facebook.html 13

7.1.3 Medium Dataset information – http://snap.stanford.edu/data/egonets-Twitter.html 14

7.1.4 Large Dataset information – http://snap.stanford.edu/data/egonets-Gplus.html 14

7.2 Citations / References 14

1 Case Study Requirements

Determine the qualitative nature of the networks you are studying and write up a report.

Do some of the following:

1. For the directed networks, sketch the **Broder et al** picture of the network.

Number of nodes in *(SCC)* strongest connected components, IN, OUT, and other sections of the network.

2. Fit a line to a log-log plot of the degree distribution.

Compute the slope of this line to determine the parameter β of the power-law degree distribution model.

3. Simulate the Price model to generate a network of n nodes for a particular power-law parameter α where $\alpha = \beta - 1$.

Compute all of the above parameters for the **Price model.**

Which (real-life) network/s does the Price's Model best represent?

1.1 Understanding of Requirements from Class Discussions

1. Choose three directed networks from http://snap.stanford.edu/data/.

2. Generate Broder et al : not directly possible using NetworkX and python, thus need to calculate number of nodes exist as SCC, IN, OUT, Tendrils, Tubes and Disconnected.

3. Plot the results for each [$x - y$] → [*log(rank) – log (deg)*].

Compute the slope of the line plotted to get parameter $\boldsymbol{\beta}$ of the power-law degree distribution model. We know that the slope $\boldsymbol{m = -\beta}$, thus $\boldsymbol{\beta = -m}$ (slope x -1)

> *Aside:* *For website analysis, say we have* $\boldsymbol{n}$ = *number of web-sites visited by* $\boldsymbol{u}$ *users. Then applying power-law, we have* $\boldsymbol{n = C\, u^{-\alpha}} \rightarrow \boldsymbol{\log(n) = \log(C) - \alpha \log(u).}$ *Thus a power-law with exponent* $\boldsymbol{\alpha}$ *is represented as a straight line having a slope* $\boldsymbol{-\alpha}$*.*

Calculate power-law (ultimately $\boldsymbol{\beta}$) for ***rank -v- in-degree*** and for ***rank -v- out-degree*** for each dataset.

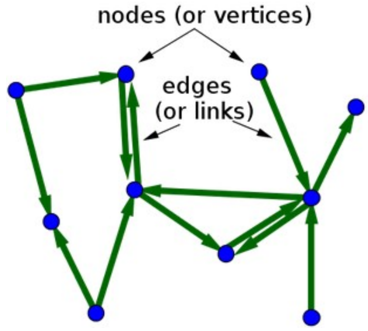
4. Simulate the Price's Model to generate a network of n nodes for power-law parameter $\boldsymbol{\alpha}$ where $\boldsymbol{\alpha = \beta - 1}$, and calculate degree distribution for Price's Model also.

5. Which real-life networks best matches the Price's Model?

2 INTRODUCTION

2.1 WHAT IS A DIRECTED NETWORK?

Source : http://mathinsight.org/directed_graph_definition



A directed graph is graph, consisting of a set of nodes (aka. vertices) that are connected together. The connections are directed from one node to another. A directed graph is sometimes called a digraph or a directed network.

In mathematical terms, this is represented by ***G* = (*V,A*),** where :

- **V** → set, whose elements are called vertices or nodes,
- **A** → set of ordered pairs of vertices, called arcs, directed edges, or arrows (and sometimes simply edges with the corresponding set named E instead of A).

2.2 DIRECTED GRAPHS USED

Each of the directed graphs chosen deal with Social media and vary in the number of nodes present. For more information on each, please see the section : References – Datasets

Small	→ ≤ 10k nodes	http://snap.stanford.edu/data/egonets-Facebook.html
Medium	→ 10k to 100k nodes	http://snap.stanford.edu/data/egonets-Twitter.html
Large	→ > 100k nodes	http://snap.stanford.edu/data/egonets-Gplus.html

3 BRODER ET AL

3.1 OVERVIEW OF BRODER ET AL

In 2000, Broder and various colleagues conducted a large-scale analysis of the web. They concluded that as a directed graph there was a recognisable structure in place that of a bow-tie (see following figure.) In short, if a page x is connected to page y via a directed edge then a hyper-link connects page x to page y.

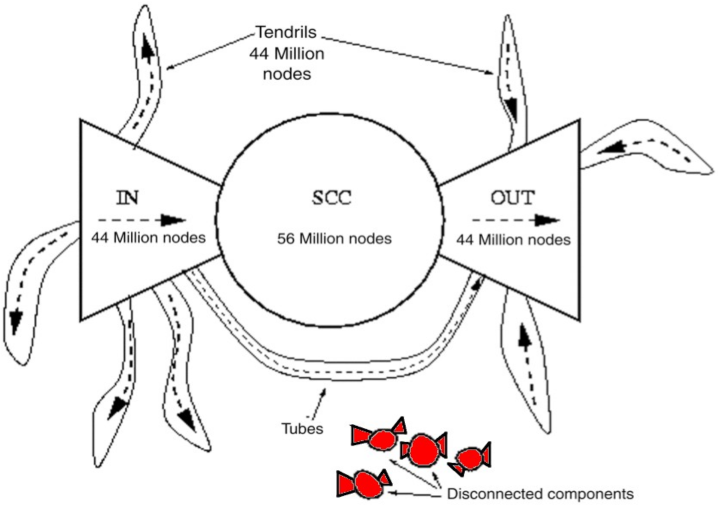


Figure 1 : Bow-tie model of the web graph (Broder et al, 2000).

3.1.1 Components : Bow-Tie Model

Component	Explanation
SCC	Nodes which are part of the 'strongly connected component' (SCC). Each nodes has *IN→edges* from IN, SCC and *OUT←edges* to OUT or SCC. Each node in SCC has one or more paths to every other node in SCC.
IN (→)	Nodes that are ***up-***stream of the SCC (i.e.: these nodes have *IN→edges* from other IN only and *OUT←edges* to other nodes in IN, SCC, Tendrils or Tubes).
OUT (←)	Nodes that are ***down-***stream of the SCC (i.e.: these nodes have *OUT←edges* to other OUT only and *IN→edges* from other nodes in OUT, SCC, Tendrils or Tubes).
Tubes	Nodes that have *IN→edges* from IN or other pages in Tubes and *OUT←edges* to nodes in Tubes or OUT.
Disconnected	Nodes that are basically outside of the bow-tie structure. Nodes have no *IN→edges* from and no *OUT←edges* to any components. These pages may be linked to each other.
Tendrils	Nodes that are can accessible from IN only or only have *OUT-edges* to OUT. These nodes do not pass through SCC.

3.1.2 How Calculated Using Python

There is no one function in NetworkX or in python to evaluate a directed graph and confirm what node is part of what component, therefore the nodes and edges must be examined to see what connections exist and in what direction.

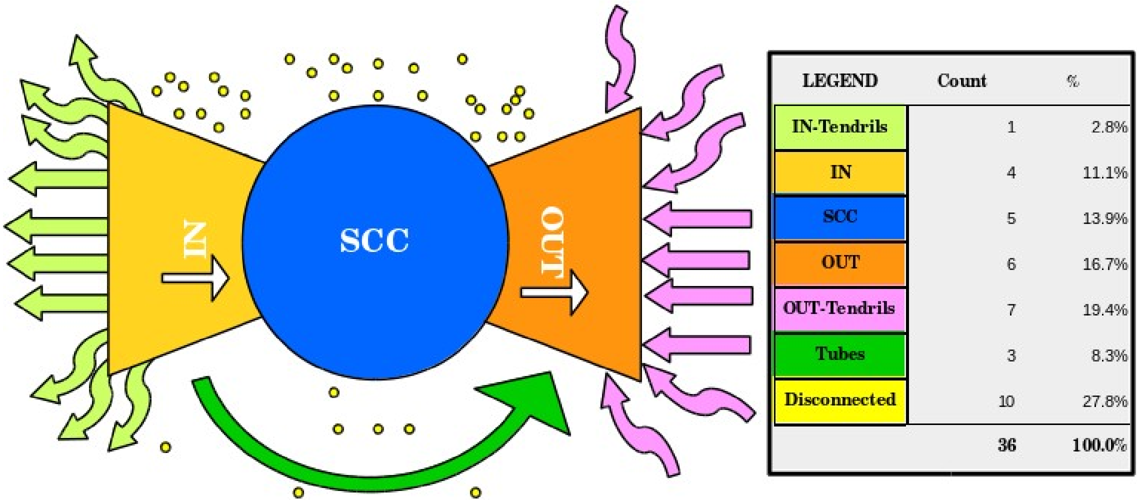
Component	Explanation
SCC	Use NetworkX function : `strongly_connected_components(G)` which generates the nodes in strongly connected component of the graph provided as a list of nodes. This sub-graph ***SCC*** **[1]** is then used as the source for ***IN*** and for ***OUT.***
IN	Use NetworkX function : *`all_pairs_shortest_path_length(G)`* which computes the shortest path lengths between all nodes in G. ***IN*** nodes will have the longest path of all nodes. **[2]**
OUT	Once IN is excluded from SCC sub-graph, nodes with shortest path are the ***OUT*** nodes. The NetworkX function : *`shortest_path(G)`* is used to calculate this. **[3]** SCC now excludes IN and OUT nodes. **[3]**
	Remaining nodes are used to calculate the remaining sets :
Tubes	IN Nodes which have a path to OUT nodes but which do not intersect SCC nodes. The NetworkX function : *`shortest_path(G)`* is used to calculate this, combined with all nodes in OUT. **[4]**
IN-Tendrils	IN Nodes which do not intersect with OUT nodes not SCC nodes. Thus if nodes are part of IN nodes but are not part of SCC (already excluded) and not part of Tubes (just calculated) then the nodes is IN-Tendril. **[5]**
OUT-Tendrils	Nodes which which flow into OUT but do not connect to SCC directly. **[6]**
Disconnected	All remaining nodes. ***[7]*** Alternatively, could use density = 0, i.e.: no connections.

The following is how Broder is calculated using python, calculation of each component is referenced from ***[1]*** to ***[7]*** :

```
def calc_broder_values(g):
    logging.info(cs_ref, 'calculate values for Broder bow-tie')
    func_intro = "\nCALCULATE BRODER  BOW-TIE VALUES ... \n"
    print func_intro
    with open(dest_file, "a") as dat_file:
        dat_file.write(func_intro)

    bt_dict = {}
    cc = nx.strongly_connected_components(g)
    lc = g.subgraph(cc.next())
    scc = set(lc.nodes())                               [1]

    shortest_path = nx.all_pairs_shortest_path_length()
    inc = {n for n in g.nodes() if scc in shortest_path[n]}
    inc -= scc                                          [2]

    outc = set()
    for n in scc:
        outc |= set(shortest_path[n].keys())
    outc -= scc                                         [3]

    tubes = set()
    in_tendrils = set()
    out_tendrils = set()
    disconnected = set()
    remainder = set(g.nodes()) - scc - inc – outc

    inc_out = set()

    for n in scc:
        inc_out |= set(shortest_path[n].keys())
    inc_out = inc_out - scc - inc - outc
    for n in remainder:
        if n in inc_out:
            if set(shortest_path[n].keys()) && outc:
                tubes.add(n)                            [4]
            else:
                in_tendrils.add(n)                      [5]
        elif set(shortest_path[n].keys()) & outc:
            out_tendrils.add(n)                         [6]
        else:
            disconnected.add(n)                         [7]

    bt_dict.update({'IN-tendrils': in_tendrils})
    bt_dict.update({'IN': inc})
    bt_dict.update({'SCC': scc})
    bt_dict.update({'OUT': outc})
    bt_dict.update({'OUT-tendrils': out_tendrils})
    bt_dict.update({'Tubes': tubes})
    bt_dict.update({'Disconnected': disconnected})

    return bt_dict
```

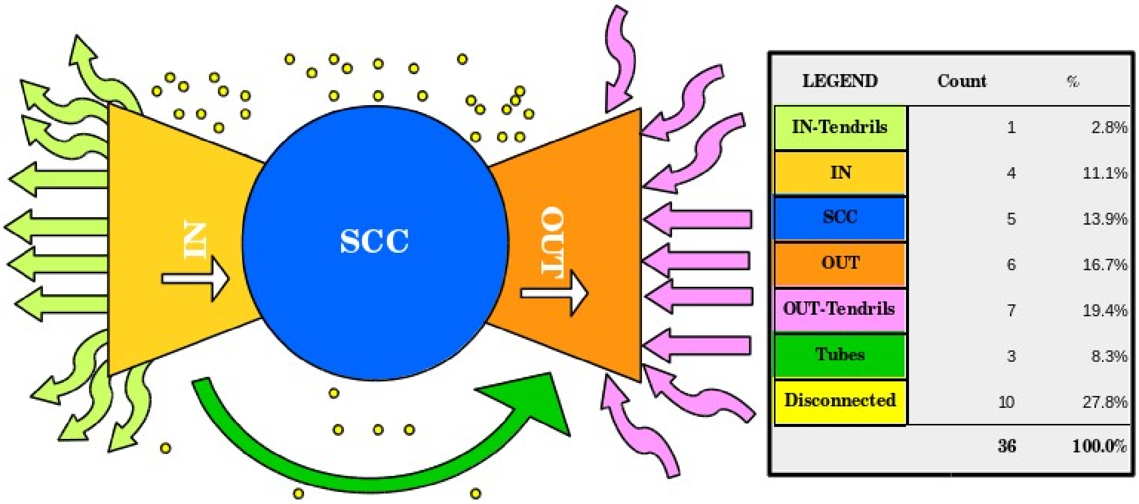
3.2 Results of Broder et al

3.2.1 Small graph → ≤ 10k nodes



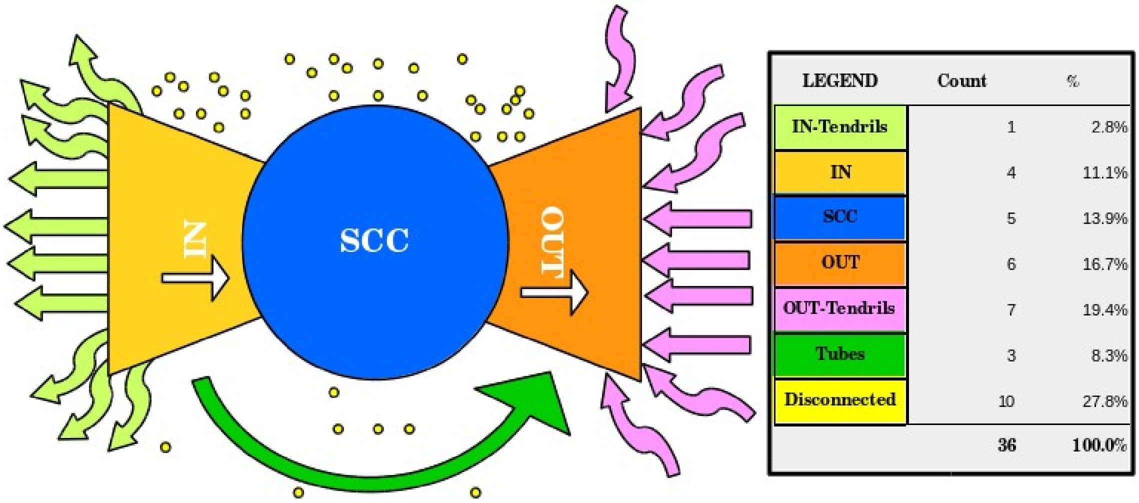
LEGEND	Count	%
IN-Tendrils	1	2.8%
IN	4	11.1%
SCC	5	13.9%
OUT	6	16.7%
OUT-Tendrils	7	19.4%
Tubes	3	8.3%
Disconnected	10	27.8%
	36	**100.0%**

3.2.2 Medium graph→ 10k to 100k nodes



LEGEND	Count	%
IN-Tendrils	1	2.8%
IN	4	11.1%
SCC	5	13.9%
OUT	6	16.7%
OUT-Tendrils	7	19.4%
Tubes	3	8.3%
Disconnected	10	27.8%
	36	**100.0%**

3.2.3 Large graph → > 100k nodes



LEGEND	Count	%
IN-Tendrils	1	2.8%
IN	4	11.1%
SCC	5	13.9%
OUT	6	16.7%
OUT-Tendrils	7	19.4%
Tubes	3	8.3%
Disconnected	10	27.8%
	36	**100.0%**

4 Plot the Degree Distribution

4.1 Overview

A power-law is a direct relationship between the rank and degree or the number of edges and the number of nodes in the directed graphs chosen. Here, we evaluate the degree-distribution for each graph.

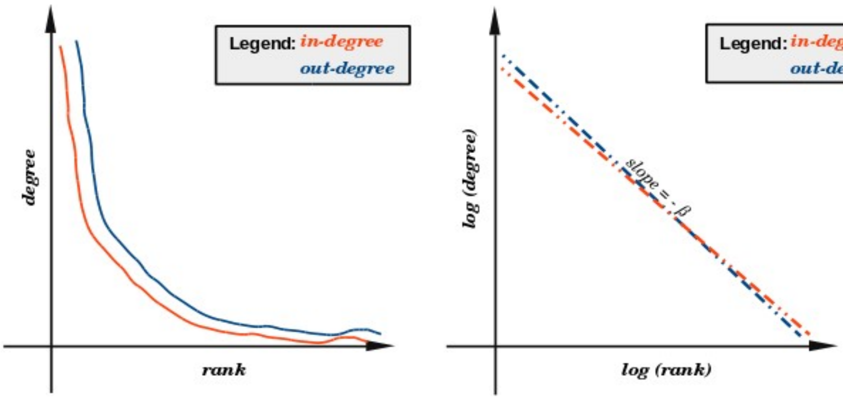


Figure 2: Power-Law Distribution for in-degree and out-degree.

4.1.1 Components : Log-Log Plot

Component	Explanation
in-degree	The number of edges ending at a node is the in-degree of that node.
out-degree	The number of edges originating at a node is the out-degree of that node.
parameter ***β***	We know that the slope ***m* = –*β***, thus ***β* = –*m*** (slope x -1)

4.1.2 How Calculated Using Python

Component	Explanation

in-degree	First calculate number of sorted in-degrees and the count of each in our graph G:
	`in_degrees = G.in_degree()`
	`in_values = sorted(set(in_degrees.values()))`
	`in_rank = [in_degrees.values().count(x) for x in in_values]`

out-degree	And also, calculate number of sorted out-degrees and the count of each in our graph G:
	`out_degrees = G.out_degree()`
	`out_values = sorted(set(out_degrees.values()))`
	`out_rank = [out_degrees.values().count(x) for x in out_values]`

Slope	Then calculate the slope ***m,*** I initially tried to use ***numpy.polyfit*** for all (x,y) for in_degree and out_degree but kept getting errors related to division by zero, so I removed (x, y) at position zero:
	`xi = in_values`
	`xi.pop(0)`
	`yi = in_rank`
	`yi.pop(0)`
	`slope, intercept = np.polyfit(np.log(xi), np.log(yi), 1)`
	I decided not to plot the line for the slope as it cause the detailed graph to become even more cluttered and perhaps illegible.

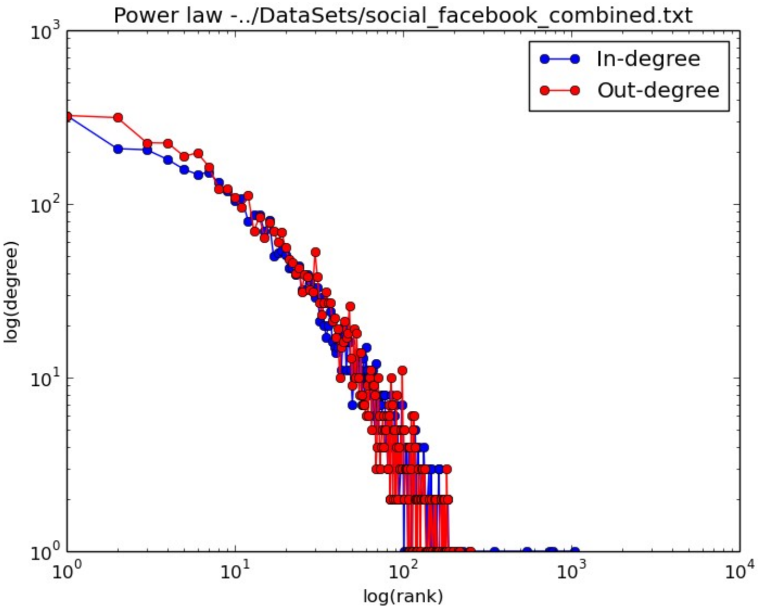
Component	Explanation

Plot results	Then for in-degree and out-degree, plot the results :
	`plt.figure()`
	`plt.loglog(out_values, out_rank, 'bv-')`
	`plt.loglog(in_values,in_rank,'ro-')`
	`plt.legend(['In-degree', 'Out-degree'])`
	`plt.title('Power law' + src_file)`
	`plt.xlabel('log(degree)')`
	`plt.ylabel('log(rank)')`
	`plt.savefig('plots/power_law.png')`
	`plt.close()`

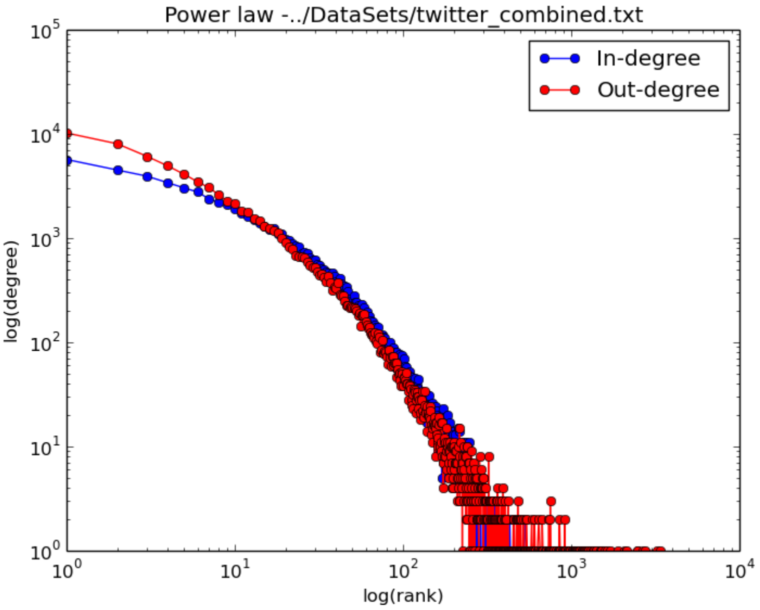
parameter ***β***	Ultimately ***β*** was calculated using : `beta = slope * -1`

4.2 Results

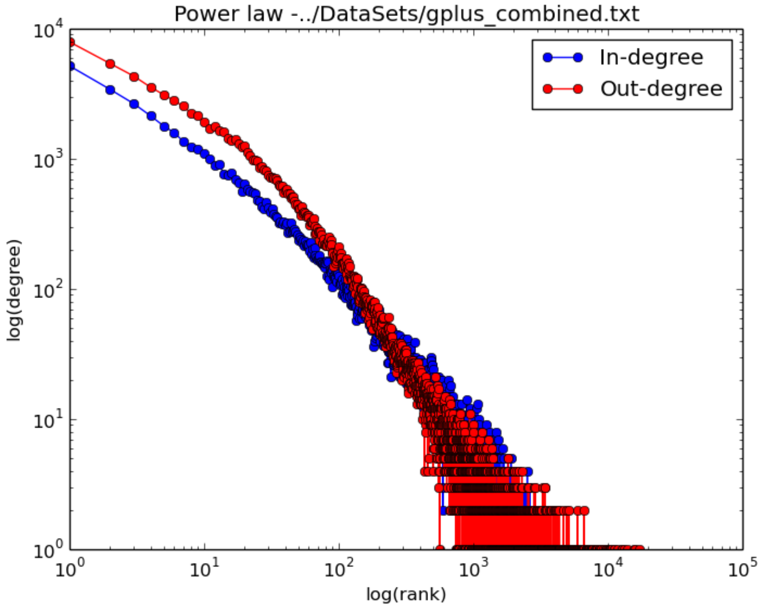
4.2.1 Small graph → ≤ 10k nodes : Power Law Distribution



4.2.2 Medium graph→ 10k to 100k nodes : Power Law Distribution



4.2.3 Large graph → > 100k nodes : Power Law Distribution



4.2.4 Parameter b for Power-Law Degree Distribution Model

Graph Details		Slope	Value of β
Small (≤ 10k) : Facebook	in-degrees	-1.47695159255	1.47695159255
	out-degrees	-1.32682677798	1.32682677798
Medium (10k to 100k) : Twitter	in-degrees	-1.74631831806	1.74631831806
	out-degrees	-2.05677900841	2.05677900841
Large (> 100k) : Google +	in-degrees	-1.3051390885	1.3051390885
	out-degrees	-1.37524044036	1.37524044036

5 Price's Model

5.1 Overview

While the Broder et al Model evaluates the physical properties of the three directed graphs chosen, Price's Model (1976) evaluates how social networks grow. Derek de Solla Price used published academic papers as his nodes. He proposed that the increase in citations by other authors should be proportional to the number of citations the paper already has. He did add a constant factor for a newly published paper which initially will have zero citations. This is basically preferential advantage → '*the rich get richer*'. This is also the first known example of a scale-free network, where we expect that regardless of the number of nodes in the graph the power-law remains constant.

5.1.1 Components : Price's Model

Component	Explanation
citations	How many times a paper was cited (referred to) by authors of other papers.
Age of paper	A new paper has 0 citations and this

5.1.2 How Calculated Using Python

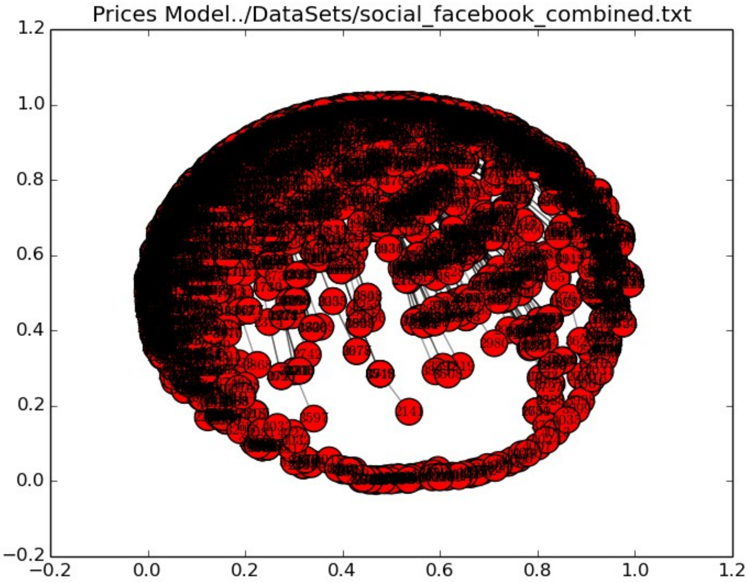
Component	Explanation
nodes	Add the same number of nodes as per the datasets used for Broder and Degree distribution.
edges	in-degrees → how many times this node(paper) was 'cited' out-degrees → how many times this node (paper) 'cited' other notes (papers) Both in-degrees and out-degrees were emulated. The resulting graphs were mapped using Matlab.

I used the same number of nodes as the actual datasets but used Price's Model to generate edges and thus had difference in-degrees and out-degrees.

5.2 Results

5.2.1 Small graph → ≤ 10k nodes : Price's Model

Created using ***n_max* = *4,039*** (number of nodes per SNAP)



5.2.2 Medium graph→ 10k to 100k nodes : Price's Model

Created using ***n_max* = *81,306*** (number of nodes per SNAP)

5.2.3 Large graph → > 100k nodes : Price's Model

Created using ***n_max* =** (number of nodes per SNAP)

5.2.4 Alpha (a) : Price's Model

Alpha (**α**) as calculated earlier using the degree distribution for the actual directed graphs was < 2 in all instances, thus a value of 3 was used for Price's Model computations. This produced the following Slope and Beta (**β**) results :

Graph Details		**Slope**	**Value of β**
Small (≤ 10k) : Facebook	in-degrees	-1.63345556109	1.63345556109
	out-degrees	-1.63345556109	1.63345556109
Medium (10k to 100k) : Twitter	in-degrees		
	out-degrees		
Large (> 100k) : Google +	in-degrees		
	out-degrees		

5.2.5 Networks best represented by Price's Model

Price's Model best illustrates networks whose degree distribution follows a power-law. Basically, the number of connections a node increases exponentially provided that node has a large number of connections (edges) initially compared to other nodes. A typical *"rich get richer"* scenario. Internet and social networks behave in a similar manner. If a web-page has a high number of hits (is deemed popular) then the number of hits will continue to grow. Conversely, if a web-site has a low number of hits (is unpopular) it will remain so.

A straight-line or a good facsimile of one strongly suggests that a power-law applies to the dataset. The slope of line equates to the power-law exponent (alpha → α). This value is ideally in the range : $2 \le \alpha \le 3$, although sometimes α lies slightly outside this range. This is observed on all datasets and price's model simulations as all points "converged" on a straight-line observed in the data plots.

The three datasets chosen (facebook, twitter and Google +) are all social networks and are all public. Price's Model usually. A social network consists of nodes; companies or individuals (commercial or personal) and some connection between them.

6 Conclusions

Broder et al and Price's Model, combined with Degree Distribution, all review components of social networks and the interaction between those components. Border et al evaluates the structure of those components. In 2000, when Broder and associates reviewed 200 million nodes, they concluded that there were at most 16 edges ("clicks") between one node and any other. Price's Model exhibit heavy-tailed distribution where popularity of a web-page increases exponentially. The Degree Distribution calculates the exponent Alpha ***(α)*** of the model using the slope of the model, where :

Alpha ***(α)*** = Beta ***(β)*** – 1 where Beta ***(β)*** = Slope x – 1.

7 REFERENCES – DATASETS

7.1 NETWORKS DATASETS

Datasets : http://snap.stanford.edu/data/index.html
The information in this section is taken directly from http://snap.stanford.edu/ and is included for informational purposes only.

7.1.1 DATASET STATISTICS – DATASETS

Details	≤ 10k nodes Facebook	10k → 100k nodes Twitter	> 100k nodes Google +
Nodes	4,039	81,306	107,614
Edges	88,234	1,768,149	13,673,453
Nodes in largest WCC	4,039 (1.000)	81,306 (1.000)	107,614 (1.000)
Edges in largest WCC	88,234 (1.000)	1,768,149 (1.000)	13,673,453 (1.000)
Nodes in largest SCC	4,039 (1.000)	68,413 (0.841)	69,501 (0.646)
Edges in largest SCC	88,234 (1.000)	1,685,163 (0.953)	9,168,660 (0.671)
Average clustering coefficient (ACE)	0.6055	0.5653	0.4901
Number of triangles	1,612,010	13,082,506	1,073,677,742
Fraction of closed triangles	0.2647	0.06415	0.6552
Diameter (longest shortest path)	8	7	6
90-percentile effective diameter	4.7	4.5	3

7.1.2 SMALL DATASET INFORMATION – HTTP://SNAP.STANFORD.EDU/DATA/EGONETS-FACEBOOK.HTML

This dataset consists of 'circles' (or 'friends lists') from Facebook. Facebook data was collected from survey participants using this Facebook app. The dataset includes node features (profiles), circles, and ego networks.

Facebook data has been anonymized by replacing the Facebook-internal ids for each user with a new value. Also, while feature vectors from this dataset have been provided, the interpretation of those features has been obscured. For instance, where the original dataset may have contained a feature "political=Democratic Party", the new data would simply contain "political=anonymized feature 1". Thus, using the anonymized data it is possible to determine whether two users have the same political affiliations, but not what their individual political affiliations represent.

Data is also available from Google+ and Twitter.

7.1.2.1 Source (citation)

J. McAuley and J. Leskovec. Learning to Discover Social Circles in Ego Networks. NIPS, 2012.

7.1.2.2 Files

File	Description
facebook_combined.txt.gz	Edges from all egonets combined

7.1.3 Medium Dataset information – http://snap.stanford.edu/data/egonets-Twitter.html

This dataset consists of 'circles' (or 'lists') from Twitter. Twitter data was crawled from public sources. The dataset includes node features (profiles), circles, and ego networks. Data is also available from Facebook and Google+.

7.1.3.1 Source (citation)

J. McAuley and J. Leskovec. Learning to Discover Social Circles in Ego Networks. NIPS, 2012.

7.1.3.2 Files

File	Description
twitter.tar.gz	Twitter data (973 networks)
twitter_combined.txt.gz	Edges from all egonets combined
readme-Ego.txt	Description of files

7.1.4 Large Dataset information – http://snap.stanford.edu/data/egonets-Gplus.html

This dataset consists of 'circles' from Google+. Google+ data was collected from users who had manually shared their circles using the 'share circle' feature. The dataset includes node features (profiles), circles, and ego networks. Data is also available from Facebook and Twitter.

7.1.4.1 Source (citation)

J. McAuley and J. Leskovec. Learning to Discover Social Circles in Ego Networks. NIPS, 2012.

7.1.4.2 Files

File	Description
gplus_combined.txt.gz	Edges from all egonets combined

7.2 *Citations / References*

Reference	Acknowledgement
SNAP → datasets	Sourced from : http://snap.stanford.edu/data
SNAP → modularity	
Price's Model	Influenced by Dr.Neil Hurley
Broder et al	Influenced by https://github.com/lamda/bowtie