Mohamed Aturban (maturban)
CS595 — FAll 2013

Assignment 4

October 10, 2013

All files mentioned in this document should be uploaded into the *github* repository.

Problem 1

The Python program *q1.py* is the answer to question 1. The code will do the following. It starts by choosing the first 100 active links stored in the file *links.txt.* For each URI, all **distinct** outbound links from that page are extracted then stored in a separate file. For Example, `http://avimeo.com/watch.php?vid=12dd9a60d#.UkGBQQq-0Nk.twitter` is the fifth active link in *links.txt*, so the program will create a file called *f5.txt* which looks like:

```
# The first link is the main page in which other
# URIs ,starting from second link , are extracted.
http://avimeo.com/watch.php?vid=12dd9a60d#.UkGBQQq-0Nk.twitter
http://avimeo.com/index.php
http://avimeo.com/login.php
http://avimeo.com/register.php
...
```

As you can see, the first two lines are just comments describing the contents of the file. The first URI is the main page from which other links are extracted, starting from second link.

Problem 2

The main task of *q2.py* is to create a *GraphViz dot* file called *graphLinks.gv* which can be used as input to *Gephi.* The following four points briefly illustrate how the dot file is created:

- Read one file (e.g. f1.txt) of 100 files
- Extract the main URI (m) and the outbound links (*l1,l2,l3, ...*)
- Before labeling edges between m and outbound links l, check if these nodes are already given labels. In this case, they will not be offered new labels. Instead. they will keep their old labels.
- Create and label node pairs : $m \rightarrow l$ [label = "$m\#$ to $l\#$"]
- Assign labels to new nodes:
  m [label = $m\#$] or
  l [label = $l\#$]
- Repeat previous five steps for all 100 files.

Again, the output dot file is called *graphLinks.gv.*

Problem 3

After Loading the dot file *graphLinks.gv* created in problem 2 into Gephi, we get the following graph (figure 1). It is difficult to organize the graph in a certain way that shows its connectivity since there are thousands of edges.

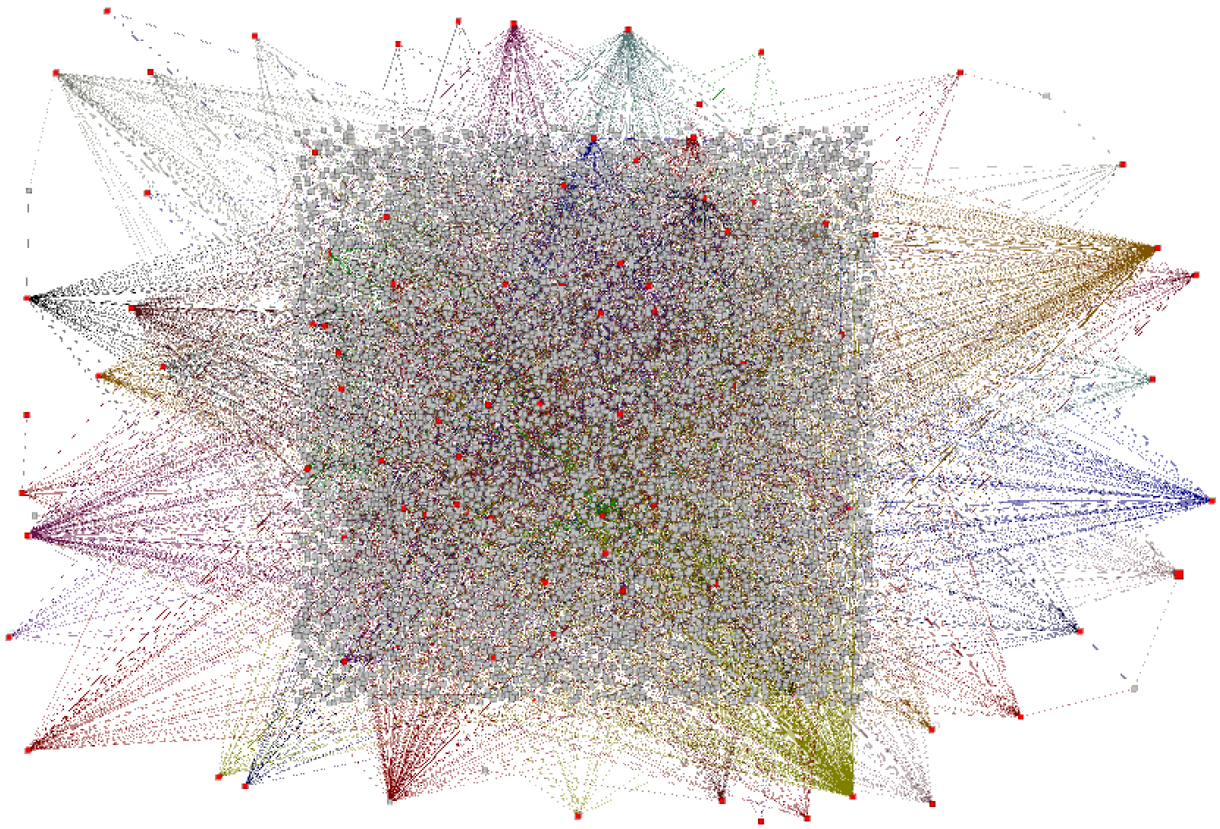

Figure 1: Graph with all nodes

Figure 2 shows examples of connected components such as:

- Node no. 10.4 is connected to main nodes: 10, 75, and 76
- Node no. 9.45 is connected to main nodes: 9 and 12

- Node no. 81.17 is connected to main nodes: 81 and 83

  ...

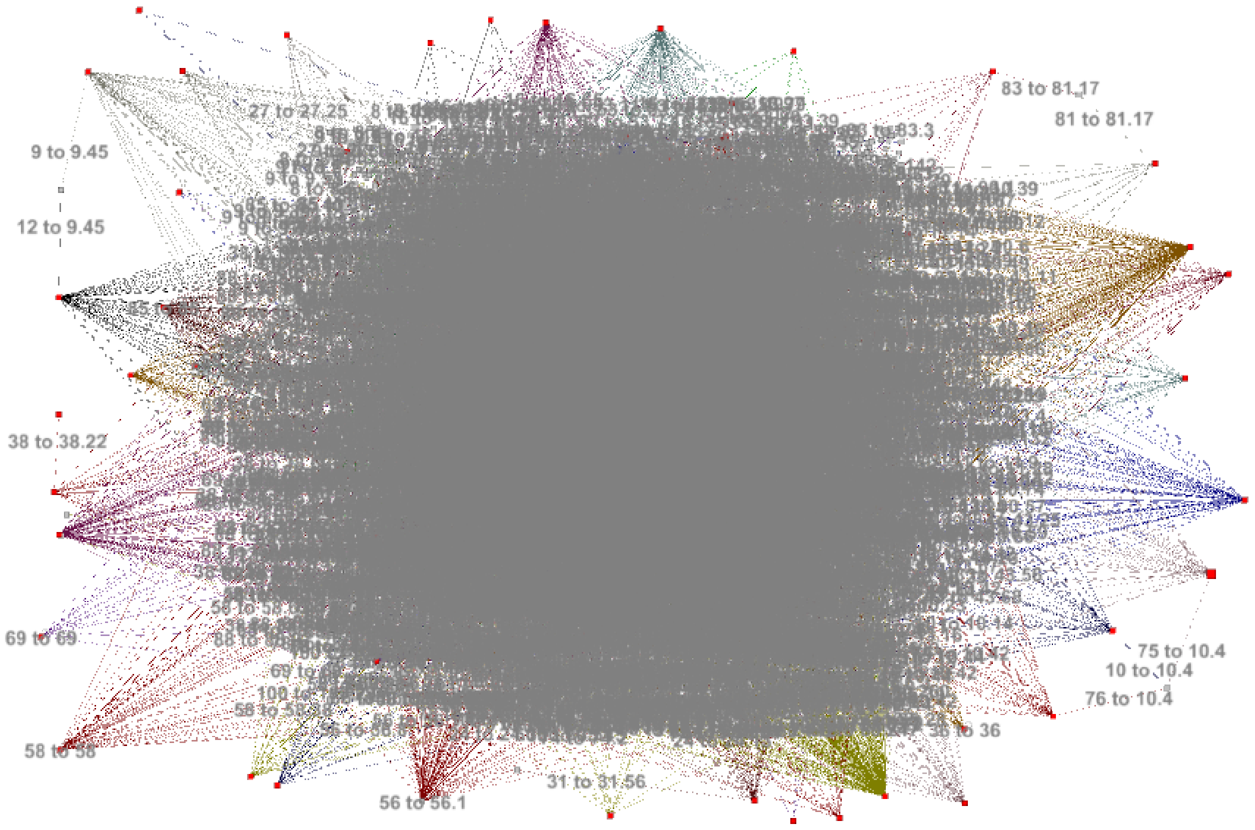

Figure 2: Graph : Connected nodes

Other statistics:

- Average Degree: 1.089
- Number of Weakly Connected Components: 58
- Number of Stronlgy Connected Components: 4577

  for more details, there is more statistical detail uploaded into the repository.

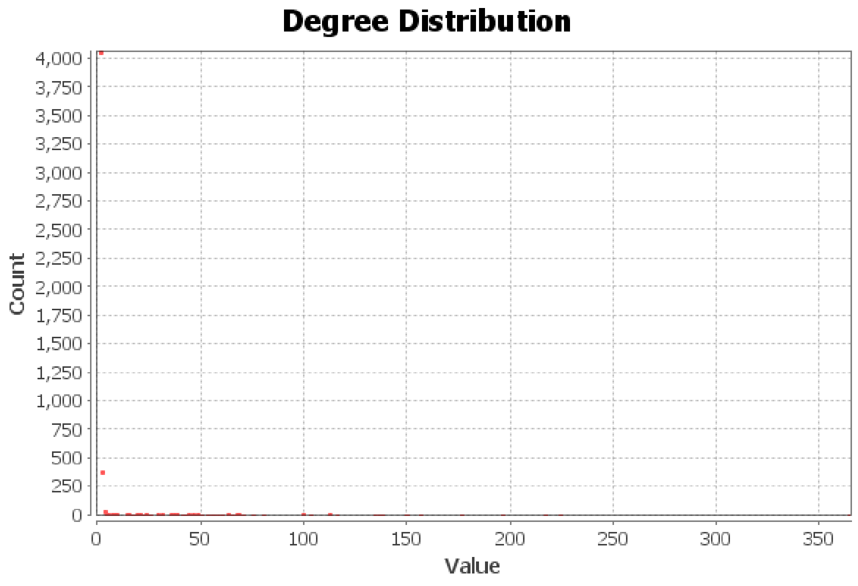

Figure 3: degree-distribution

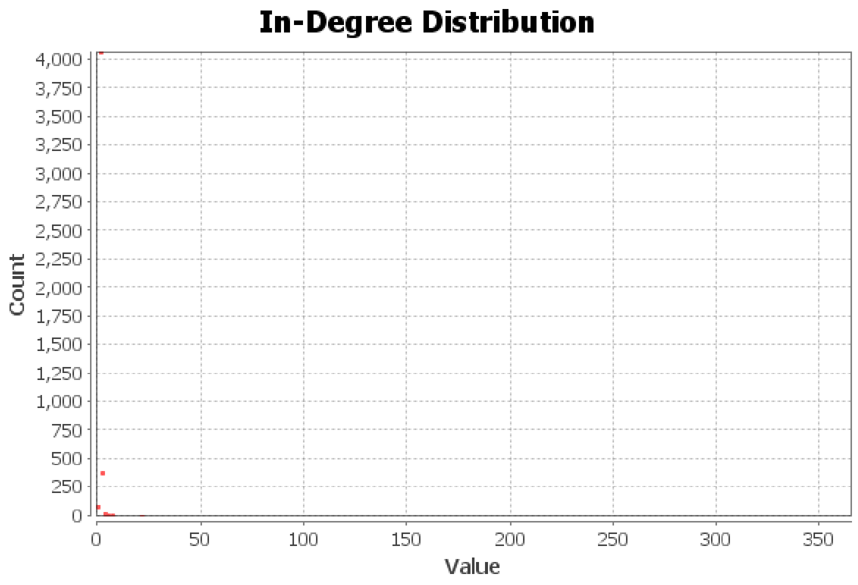

Figure 4: indegree-distribution

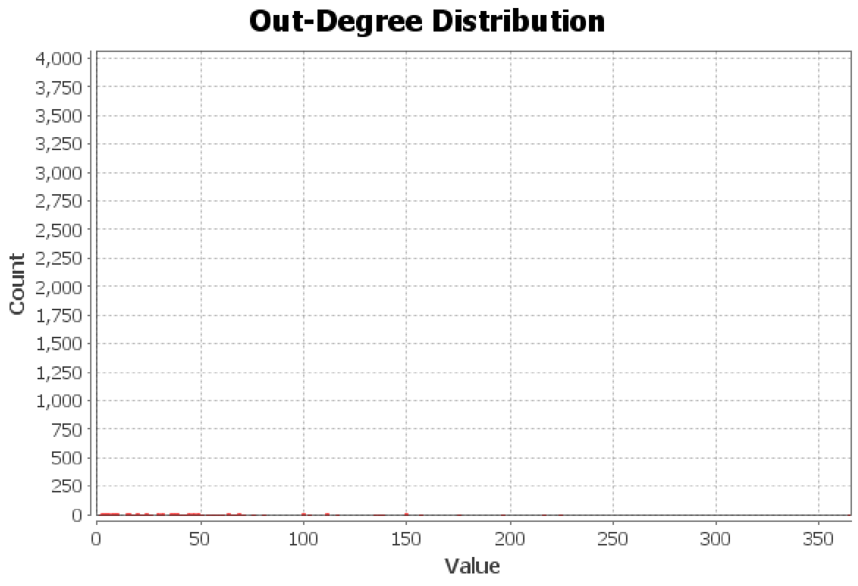

Figure 5: outdegree-distribution

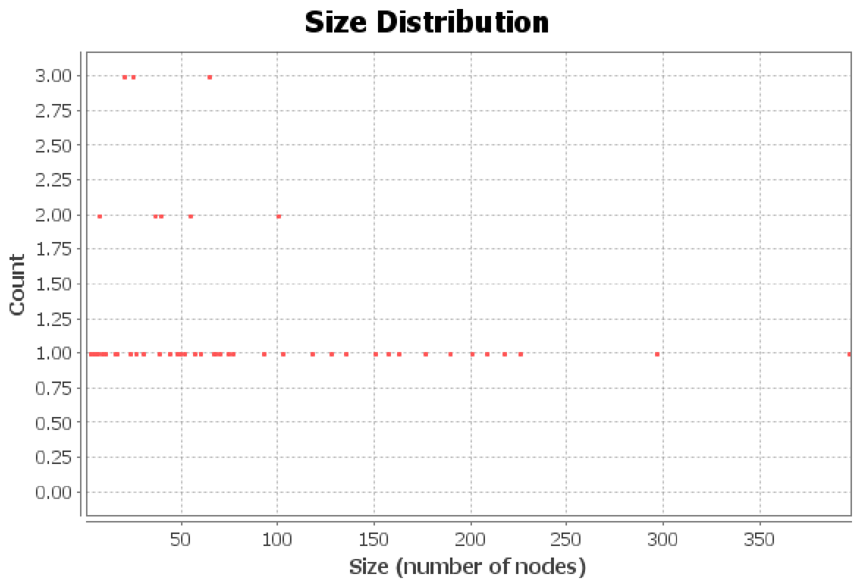

Figure 6: cc-size-distribution

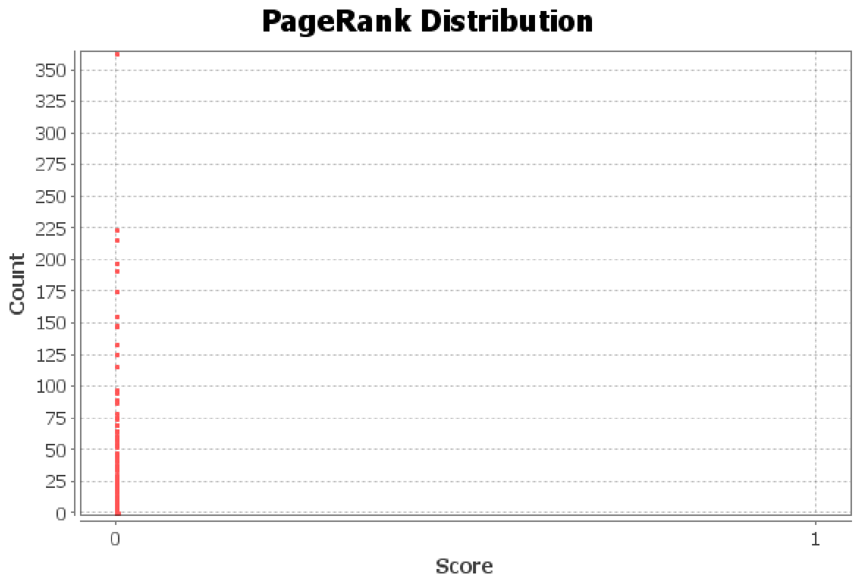

Figure 7: pageranks

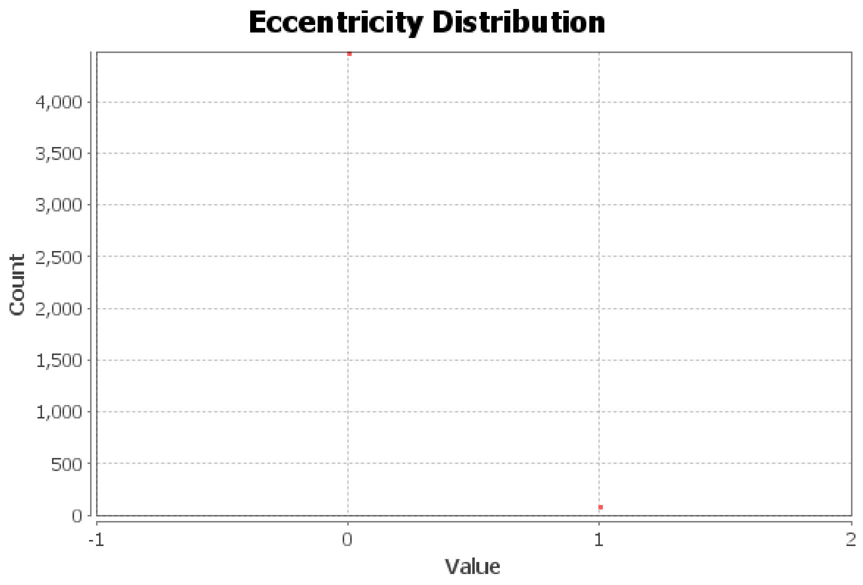

Figure 8: Eccentricity Distribution

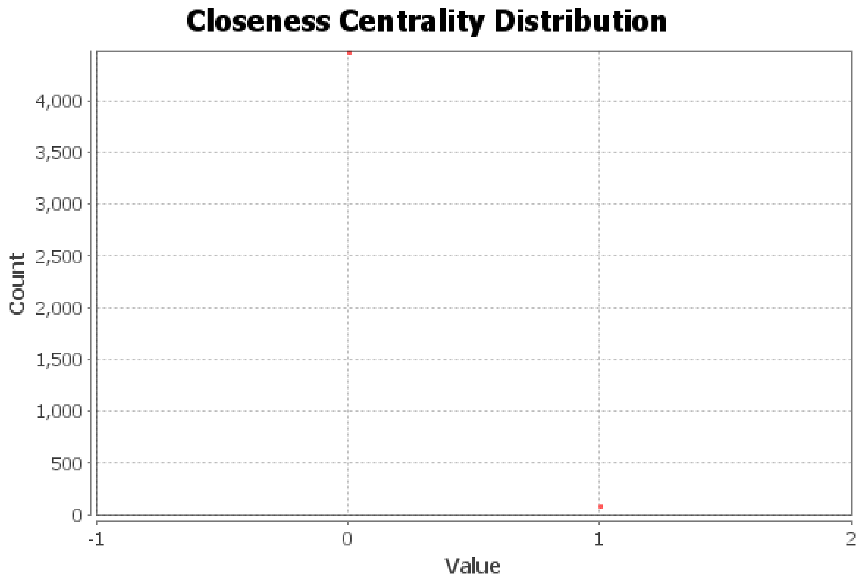

Figure 9: Closeness Centrality Distribution

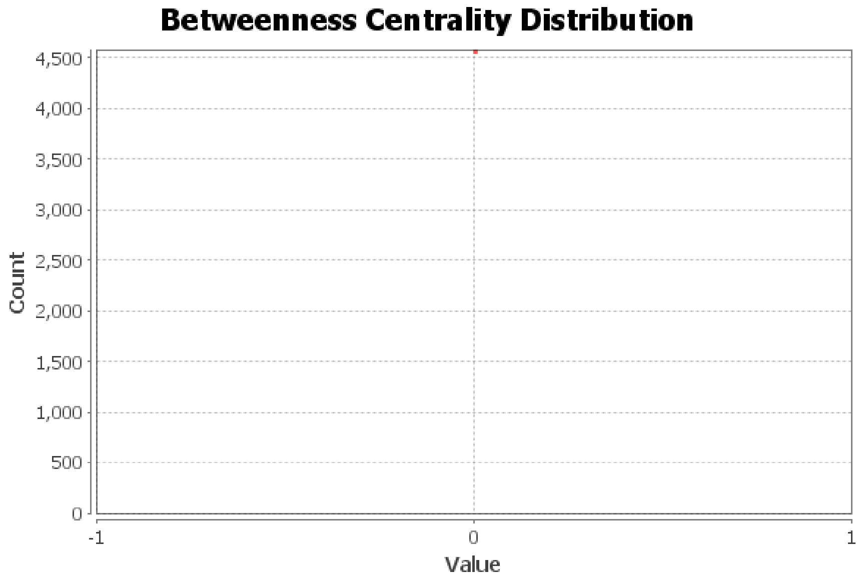

Figure 10: Betweenness Centrality Distribution