Analyzing small world phenomenon and scale free network of python packages

Avadhoot Agasti
aagasti@indiana.edu

Abhishek Gupta
abhigupt@iu.edu

ABSTRACT

The *small world* phenomenon is a very important and interesting concept which can be practically used in optimizing imports for programming languages like python, java etc. This will also help us understand the network of the packages used by developers and classify them as scale free or random network. Further, this analysis can be extended to any other programming language and understand the package network. The scope of this project is to focus on python based packages and navigate to all dependent packages to build a network graph of these packages. Further, identify the important packages which could be like important nodes in scale-free network.

KEYWORDS

scale-free, random networks, small-world, packages

1 INTRODUCTION

When try to install a python library or rather when we use this python library first thing that we need to do is we have to *pip install* it. More than often we run into an issue dependencies where its missing some dependency and when we install that dependency it complains for some other dependency and goes on till we install these dependencies. Fundamentally, its missing some core dependencies as well. Why cant' python bundle these core dependencies? But then how to identify those? The python libraries keep growing so how to we identify which one needs to added/removed/updated etc?

2 REQUIREMENTS

The goal of this analysis is

- to create a network graph of these dependencies
- understand the dependencies and important nodes
- identify if its a scale-free network
- identify any other properties depicted by this graph
- update the graph and build the graph as and when needed

At the end of this exercise we should be able to run this analysis on available python [**?**] packages. Also, we need understand and compute the other properties of the graph like average degree, average clustering co-efficient, average path length etc Also verify if the graph is *scale free* or just a *random* graph.

3 TECHNICAL SOLUTION

Our approach is to identify the dependent packages by looking at the dependencies using *pip show* output. For example

```
pip show networkx
```

```
Name: networkx

Version: 1.11

Summary: Python package for creating and
manipulating graphs and networks

Home-page: http://networkx.github.io/

Author: NetworkX Developers

Author-email: networkx-discuss@googlegroups.com

License: BSD

Location: /Libs/anaconda/lib/python3.6/site-packages

Requires: decorator
```

Looking at the above output we can clearly see that *networkxx* depends on package *decorator* and further package *decorator* may be dependent on other package and so on. We continue to traverse we should be able to find all dependent packages till we reach code python libraries. Interestingly, there *128k* python packages available [**?**] We can automate running this analysis on these packages using python script. After collecting this information each package will form a node in the graph and each edge will represent the dependency with the next module. We should be able to plot this graph using *networkx* module and represent the most important nodes which become the hubs to the network. Also, compute the degree and clustering co-efficient for this graph. Further, we can also study to graph to identify if the graph follows the *power law* distribution or it is a *scale free* network or its just a *random* network.

4 CHALLANGES

During this exercise we can run into few technical challenges for example to perform this analysis we have to install each python package locally which could be really resource intensive specially in terms of disk space. Also, while doing the analysis and loading the graph for 128k packages could be memory and cpu intensive. To get around this problem we may have to reduce the analysis to less number of python packages and keep adding mode packages as we make progress.

5 ACKNOWLEDGEMENTS

The authors thank Prof. Katy Borner for her technical guidance. The authors would also like to thank TAs of Information Visualization class for their valued support.