Community Detection on the Reddit Hyperlink Dataset

[redacted]
Department of Computer Science
University of Exeter, Exeter, UK

Internal Supervision:
[redacted]
University of Exeter, Exeter, UK

Internal Supervision:
[redacted]
University of Exeter, Exeter, UK

***Abstract*—This report covers the application of two community detection algorithms, the Louvain algorithm and the Label Propagation algorithm, when applied to a large dataset of hyperlinks between forums on the popular forum site Reddit. To this end, background is provided surrounding the Reddit Hyperlink dataset, followed by the motivation for the choice of algorithms. Following this, technical details for the Louvain and Label Propagation algorithms are provided, including explanations of their key algorithmic components and time complexity.**

Further, this report evaluates and compares the two algorithms using visual representations. The two sets of communities produced by both algorithms are also evaluated based on their Triangle-Participation Ratio, Conductance and Network Modularity values, which display scoring based on internal connectivity, combined connectivity and the entire network model respectively.

I. Introduction: Contextualization and Motivation

In this report, I detail the use of community detection algorithms when applied to the Reddit Hyperlink Dataset [1], a dataset of connections between established communities on the popular forum-based website Reddit. Doing this, I apply and compare two different community detection algorithms: the Louvain algorithm [2] and the Label Propagation algorithm [3] (henceforth referred to as the LPA).

A. The Reddit Hyperlink Dataset

For this mini-project I have chosen to use the Reddit Hyperlink Dataset [1], hosted on the Stanford Large Network Dataset Collection [4]. The Reddit Hyperlink Dataset was created by Kumar et al. [1] to study cases of 'intercommunity conflict', where members of one Reddit community (called a 'Subreddit') collectively mobilise to participate in or attack another community [1].

The dataset itself consists of a Tab-Separated Values (TSV) file containing 40 months of Reddit comments and posts extracted between January 2014 and April 2017, in which a hyperlink to another subreddit was present. As such, each entry in the dataset represents a directed edge between two subreddit nodes and allows one to study communities that may exist between various subreddits. The dataset also provides the Post ID, the timestamp of the post, a label indicating explicit sentiment towards the target subreddit and a vector representing the text properties of the source post [1].

B. Chosen Algorithms

The Reddit Hyperlink dataset [1] contains $67,180$ individual nodes with $339,643$ edges, but only 712 connected components; following further investigation, there are 711 connected components containing between 2 and 9 nodes, and a single connected component containing $65,648$ nodes. In this case community detection was carried out on this component only; the other, smaller components can be thought of as independent communities already.

In the pitch presentation, I proposed the use of three different algorithms for community detection: the Louvain Algorithm [2], the Girvan-Newman algorithm [5] and the Fluid Communities algorithm [6]. During development and testing of the code for this project, I discovered some issues with this initial plan.

The Girvan-Newman algorithm progressively removes edges from a graph of m edges and n nodes, based on an edge's betweenness centrality [7]. The algorithm runs a fast edge betweenness calculation for all m edges in the graph of n nodes, running in time $O(mn)$ [8]; but, this calculation is repeated for the removal of each edge, so the Girvan-Newman algorithm runs in time $O(m^2n)$ [5]. Due to the sheer scale of the Reddit Hyperlink Dataset [1], the Girvan-Newman algorithm did not produce results within a feasible timeframe (6 hours) when applied to the dataset and often exceeded the allocated memory of 16 GB. As such, I chose not to pursue the Girvan-Newman for this report.

The Fluid Communities algorithm (henceforth FluidC) [6] requires the specification of k communities to be found. One could perform a modularity maximisation search by running FluidC for a varying value of k, perhaps iterating within the range $2 \leq k \leq n - 1$, for n nodes and evaluating the modularity. However, the FluidC algorithm would need to run across $n - 2$, or $65,646$, iterations for full coverage. In initial tests, the computation time of FluidC, averaged across 10 runs, was 8.32 seconds which, over $65,646$ iterations, would take $546,174$ seconds, or just over 151 hours. Considering this, I chose not to pursue the FluidC algorithm in this report.

Due to these two factors, I instead chose to use the Louvain algorithm [2] and the LPA [3].

II. Methods: Description and Implementation

Before any community detection algorithms can be applied to the Reddit Hyperlink Dataset, some pre-processing must be carried out to ensure compatibility with the NetworkX package [9]. First, the original TSV file is downloaded from the Stanford Large Network Dataset Collection and converted a Comma-Separated Values (CSV) file. Following this, the CSV file is loaded into a DataFrame [10], [11] in which only the `SOURCE_SUBREDDIT` and `TARGET_SUBREDDIT` fields are extracted.

As each row represents any post or comment containing a hyperlink from the source subreddit to the target subreddit, rows can repeat source and target subreddits. The occurrences of each row are counted and then placed into a `WEIGHT` field for each edge. For example, if a row containing the source 'destinythegame' and target 'gaming' appears 5 times within the CSV file, then the weight for the edge from 'destinythegame' to 'gaming' would be 5. The weighted edgelist is subsequently saved as a separate CSV file, which is provided within the submission.

A. *Louvain Algorithm*

Introduced in 2008 by Blondel et al. [2], the Louvain algorithm finds partitions of high modularity in large networks by unfolding the network into a complete hierarchical structure [2]. The algorithm itself is divided into two phases, which are repeated iteratively; initially in phase one, each node is assigned its own community so that there are as many communities as are nodes in the network. For each node i and its neighbours j of i, the gain in modularity that might occur if i was moved from its own community to the community of j is evaluated; node i is subsequently placed into the community for which the modularity gain is a positive maximum [2]. This is repeated for every node, repeatedly until a local maxima of modularity is achieved. In the second phase of the algorithm, a new network is built whose nodes are now the communities found during phase one. The weights on edges between the new nodes are found as the sum of the weights between nodes in the corresponding two communities [12]. Once this new network is built, the first phase of the algorithm is then reapplied and iterated; in this case, a 'pass' is denoted as a combination of the two phases [2]. The passes are then iterated until no changes are made and a global maxima of modularity is attained.

To apply this algorithm to the Reddit Hyperlink Dataset [1], I use the `python-louvain` package [13]. The `community.best_partition()` function returns a dictionary containing each node and the numerical label of the community that it belongs to.

```python
import community
import pandas as pd

louvain = community.best_partition(connected)
louvain_labels = pd.DataFrame(list(louvain.items
()), columns=['node', 'community'])
```

The above code snippet shows the necessary code required to run the Louvain algorithm, where `connected` is the single connected component mentioned in Section I-B.

B. *Label Propagation Algorithm*

Introduced in 2007 by Raghavan et al. [3], the LPA is a near-linear community detection algorithm. Compared to hierarchical methods like the Walktrap algorithm, which successively groups smaller communities together based on a similarity measure again into a series of partitions [14], the LPA carries out localised detection by initialising each node in a network with a unique label and propagating labels throughout the network [3].

Supposing that a node x has k neighbours, $x_1, x_2, \dots, x_k$, and each neighbour carries a label denoting the community to which they belong. The node x determines its own community based on the labels of its neighbours, choosing to join the community to which the maximum number of its neighbours belongs [3]. Ties may occur when determining the maximum label, so they are broken uniformly randomly. For asynchronous LPA, nodes update using the rule found in Equation 1, where $C_x(t)$ is the label of node x at time t, $x_{i1}, \dots, x_{im}$ are neighbours of x that have already been updated in the current iteration, and $x_{i(m+1)}, \dots, x_{ik}$ are neighbours that are not yet updated:

$$C_x(t) = f(C_{x_{i1}}(t), \dots, C_{x_{im}}(t), C_{x_{i(m+1)}}(t-1), \dots, C_{x_{ik}}(t-1)) \tag{1}$$

The order in which all n nodes in the network are updated is chosen randomly [3]. The iterative process itself should, ideally, continue until no node changes its label; however, nodes may have an equally maximum number of neighbours in two or more communities. Due to the uniformly random tie-break, labels on such nodes can change over iterations even if the neighbouring labels remain the same. Hence, the process is carried out until every node in the network has a label to which the maximum number of its neighbours belongs; this stopping rule is shown in Equation 2.

If $C_1, \dots, C_p$ are the labels currently active in the network and $d_c^{C_j}$ is the number of neighbours node i has with nodes of label C_j, then the algorithm is stopped for every node i:

$$\text{If } i \text{ has label } C_m \text{ then } d_i^{C_m} \geq d_i^{C_j} \ \forall j \tag{2}$$

The LPA also runs in near-linear time; initialising all nodes n with a unique label requires $O(n)$ time. Alongside this, each iteration of the LPA takes linear time of $O(m)$ for m edges in the network; at each node x, picking the maximum size group and assigning a label takes worst-case time of $O(d_x)$, for d_x neighbouring labels. Further, it is possible that two or more disconnected groups obtain the same label; in such cases a simple breadth-first search is carried out which increases the overall time complexity to $O(m + n)$.

To implement the LPA, I use the NetworkX `community` module [9]. The following code snippet shows the necessary code required to run the LPA, where `connected` is the single connected component mentioned in Section I-B:

```python
from networkx import community as com

lpa = com.async_lpa_communities(connected, 'weight')
```

```python
lpa_labels = list(lpa)
```

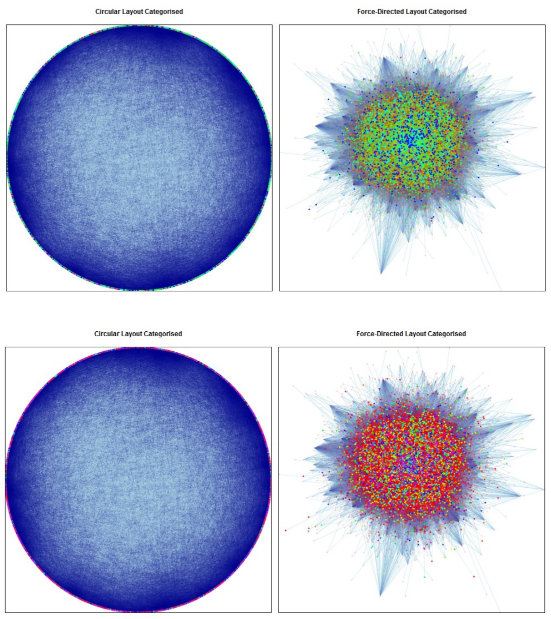
Circular Layout Categorised
Force-Directed Layout Categorised
Circular Layout Categorised
Force-Directed Layout Categorised

Fig. 1: Top: Circular and Force-Directed layout of Louvain communities; Bottom: Circular and Force-Directed layout of LPA communities

III. Experiments and Results

Figure 1 shows the Circular and Force-Directed layout of both Louvain and LPA communities. As can be seen in the Force-Directed layout, the Reddit Hyperlink Dataset [1] is a very centralised, densely clustered dataset, with a handful of nodes sitting on the outskirts of the central hub and only communicating via few, key edges and nodes.

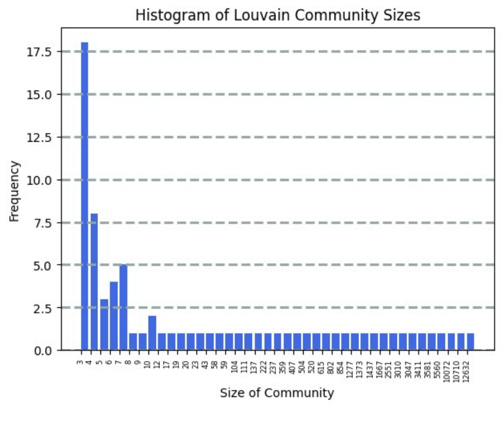

Fig. 2: Histogram of Louvain Community Sizes

As can be seen in Figures 2 and 3, the Louvain algorithm generates fewer communities than the LPA (generating 75), but generates a more even spread of sizes than the LPA; a majority of Louvain communities contain between 3 and 10 nodes. In comparison, the LPA detects more communities of smaller size, having a clear bias towards smaller communities in general; the LPA detects 1367 communities. Further, the LPA detects a single, very large community of around $58,000$ nodes suggesting that there is a large, dense, central hub of nodes through which a single label dominates.

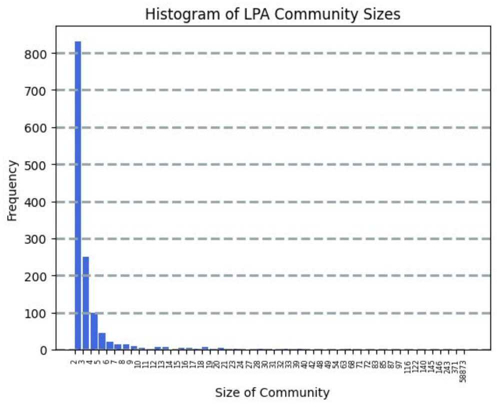

Fig. 3: Histogram of LPA Community Sizes

The Louvain communities have a minimum conductance (i.e. the fraction of total edge volume that points outside of a community [15]) of 0.003 and a maximum conductance of 0.652, whereas the LPA communities have a higher minimum and maximum conductance of 0.008 and 0.714 respectively, suggesting that the LPA finds communities that are, on average, better separated [16] than the Louvain communities.

Yang et al. determine that the Triangle Participation Ratio (henceforth TPR) is the most appropriate evaluation metric for dense, heavily overlapping networks [16]. Given this, the Louvain communities have a TPR score of 0.240 and the LPA communities have a lower TPR score of 0.04, suggesting that the Louvain algorithm detects communities that are better connected internally, with more nodes within the Louvain communities being a part of Triangles.

Finally, the Louvain algorithm produces communities with a modularity of 0.417, whereas the LPA produces communities with a much smaller modularity of 0.052, suggesting that the Louvain algorithm prefers denser communities with sparser connections to other communities. The LPA, on the other hand, seems to generate smaller, less dense communities which connect more with other communities [17].

IV. Conclusions

In this report I have discussed the Reddit Hyperlink dataset [1], which is a large dataset of hyperlinks between Subreddits on the forum site Reddit. I have also discussed the Louvain algorithm and the near-linear Label Propagation algorithm, how they operate and their implementation in Python. Further, I have showed visualisations of the network and the communities found by these algorithms, with the Louvain algorithm detecting fewer, but larger communities than the Label Propagation algorithm.

References

[1] S. Kumar, W. L. Hamilton, J. Leskovec, and D. Jurafsky, “Community interaction and conflict on the web,” in *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2018, pp. 933–943.

[2] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[3] U. N. Raghavan, R. Albert, and S. Kumara, “Near linear time algorithm to detect community structures in large-scale networks,” *Physical review E*, vol. 76, no. 3, p. 036106, 2007.

[4] J. Leskovec and A. Krevl, “SNAP Datasets: Stanford large network dataset collection,” http://snap.stanford.edu/data, Jun. 2014.

[5] M. Girvan and M. E. Newman, “Community structure in social and biological networks,” *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[6] F. Parés, D. G. Gasulla, A. Vilalta, J. Moreno, E. Ayguadé, J. Labarta, U. Cortés, and T. Suzumura, “Fluid communities: A competitive, scalable and diverse community detection algorithm,” in *Complex Networks & Their Applications VI: Proceedings of Complex Networks 2017 (The Sixth International Conference on Complex Networks and Their Applications)*. Springer, 2018, pp. 229–240.

[7] L. C. Freeman, “A set of measures of centrality based on betweenness,” *Sociometry*, pp. 35–41, 1977.

[8] M. E. Newman, “Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality,” *Physical review E*, vol. 64, no. 1, p. 016132, 2001.

[9] A. A. Hagberg, D. A. Schult, and P. J. Swart, “Exploring network structure, dynamics, and function using networkx,” in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.

[10] The Pandas Development Team, “pandas-dev/pandas: Pandas,” Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134

[11] Wes McKinney, “Data Structures for Statistical Computing in Python,” in *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman, Eds., 2010, pp. 56 – 61.

[12] A. Arenas, J. Duch, A. Fernández, and S. Gómez, “Size reduction of complex networks preserving modularity,” *New Journal of Physics*, vol. 9, no. 6, p. 176, 2007.

[13] T. Aynaud, “python-louvain x.y: Louvain algorithm for community detection,” `https://github.com/taynaud/python-louvain`, 2020.

[14] P. Pons and M. Latapy, “Computing communities in large networks using random walks,” in *Computer and Information Sciences-ISCIS 2005: 20th International Symposium, Istanbul, Turkey, October 26-28, 2005. Proceedings 20*. Springer, 2005, pp. 284–293.

[15] J. Shi and J. Malik, “Normalized cuts and image segmentation,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[16] J. Yang and J. Leskovec, “Defining and evaluating network communities based on ground-truth,” in *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, 2012, pp. 1–8.

[17] M. E. Newman and M. Girvan, “Finding and evaluating community structure in networks,” *Physical review E*, vol. 69, no. 2, p. 026113, 2004.