Anomaly Detection for Network Data Using MapReduce

Justin Conklin, Israel Showell; Dr. Enyue Lu

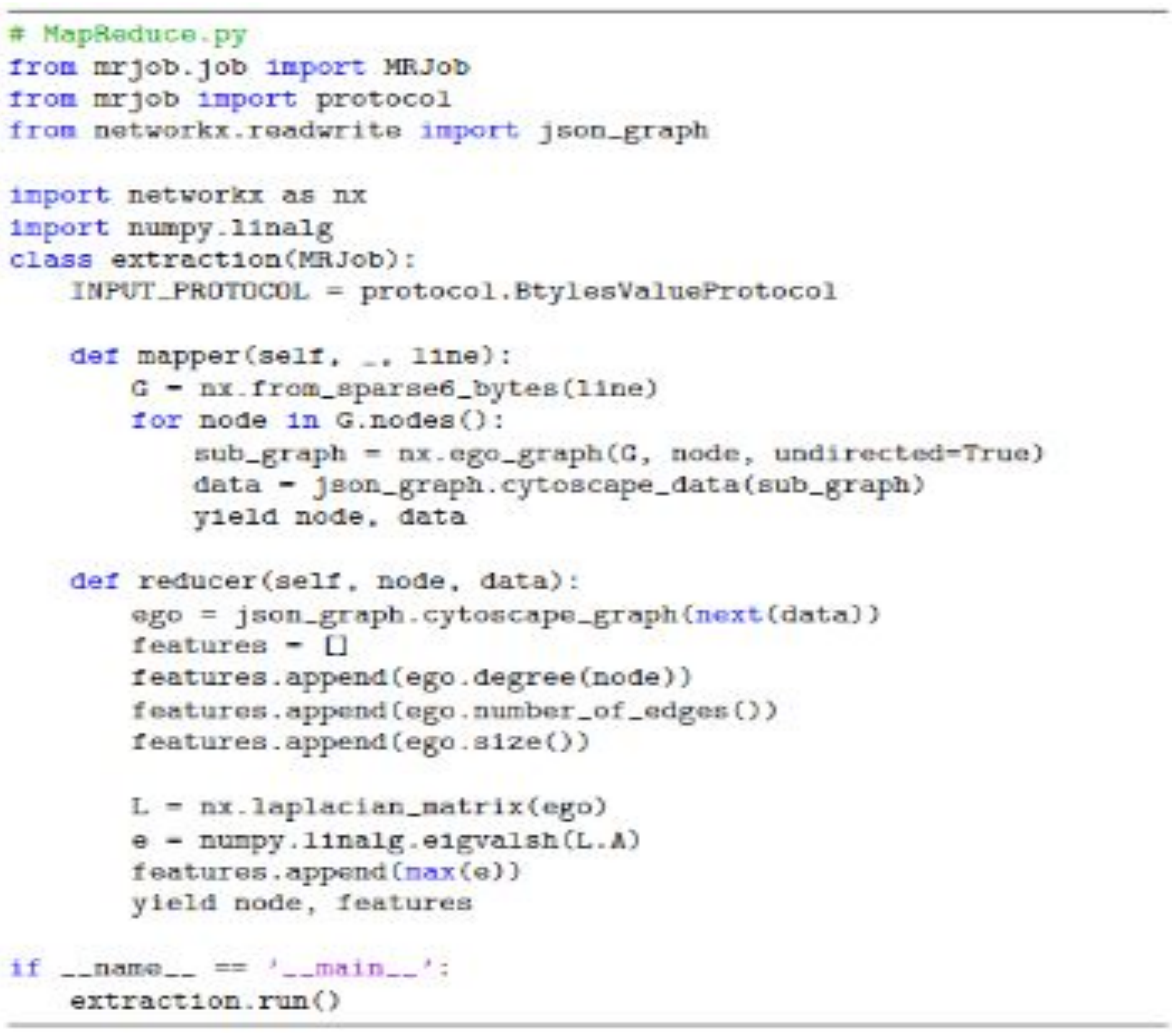
Department of Computer Science, Salisbury University; Department of Computer Science, University of Maryland Eastern Shore

Abstract

Network graphs are used to represent large amounts of data in a variety of fields. The ability to detect anomalies in network graphs has previous research in the Oddball algorithm paper. This poster extends the previously conducted research to study whether the power law observations have a correlation over a time series data-set. The benefit of this would be the ability to quickly assess network connections over real time for anomalous behavior. The findings of this research indicate that the previously observed power law relationships hold over time but have no correlation and cannot be used to improve accuracy. The MapReduce algorithm was implemented to simulate real-time server wide analysis and to handle the large amount of data and calculations needed for feature extraction.

Method

All programs are compiled utilizing Python 2 [8]. Running the program requires providing two separate files. The first file is the input file which consist of a text file containing four columns as generated through the preprocessing step. (See figure below). The second file is the list of anomalous records which is a copy of the anomalous labeled csv file available from MAWILab [2]. Using a similar method as the Oddball algorithm, an outlier score is generated for each node. This outlier score is determined by two separate variables. The first, is the distance of the data point from the line of best fit and is calculated using the euclidean distance formula. The second is determined using the Local Outlier Factor algorithm which generates a score based on the relative density of a point [3]. The sum of these two scores after they are normalized by dividing by their max value is equal to the outlier score for that node.



```python
# MapReduce.py
from mrjob.job import MRJob
from mrjob import protocol
from networkx.readwrite import json_graph

import networkx as nx
import numpy.linalg
class extraction(MRJob):
    INPUT_PROTOCOL = protocol.BtylesValueProtocol

    def mapper(self, _, line):
        G = nx.from_sparse6_bytes(line)
        for node in G.nodes():
            sub_graph = nx.ego_graph(G, node, undirected=True)
            data = json_graph.cytoscape_data(sub_graph)
            yield node, data

    def reducer(self, node, data):
        ego = json_graph.cytoscape_graph(next(data))
        features = []
        features.append(ego.degree(node))
        features.append(ego.number_of_edges())
        features.append(ego.size())

        L = nx.laplacian_matrix(ego)
        e = numpy.linalg.eigvalsh(L.A)
        features.append(max(e))
        yield node, features

if __name__ == '__main__':
    extraction.run()
```

Purpose

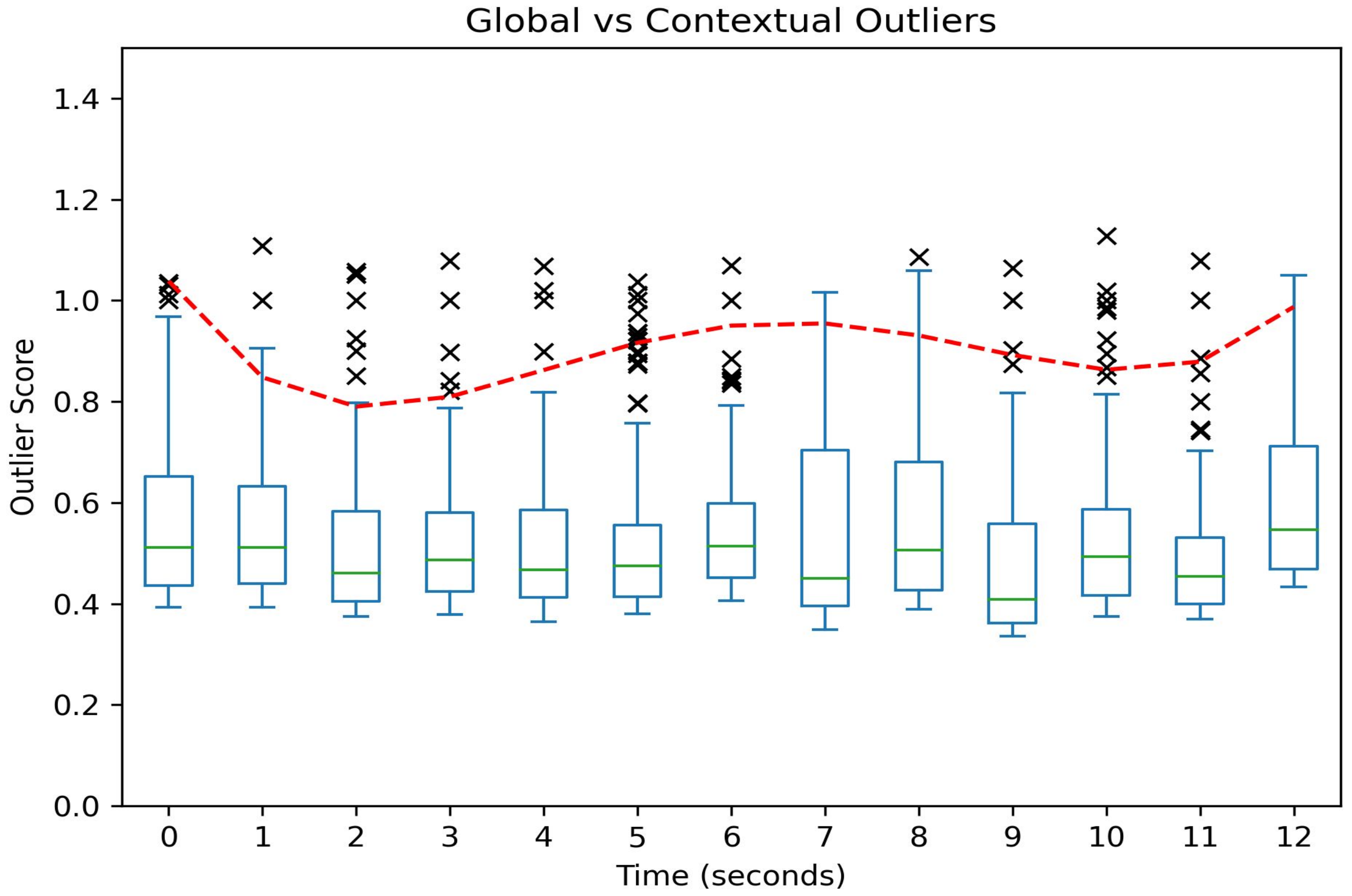
This work extends the previous works of weighted graph anomaly detection via the Oddball algorithm [1] by implementing it within a MapReduce framework and utilizing time series data. The implementation of the time series method was tested with real-world data sets and analyzed for consistency and accuracy of anomaly detection. The main result we wanted to produce was higher accuracy, faster runtimes, and better classification rates.

Results

Each grouping is placed on a plot as a box and whisker graph. A general trend line is constructed to test if there is a correlation between power law coefficients that can be used to more accurately determine anomalous behavior. Figure below shows the graph generated from the MAWILab data-set. The full labeled data-set is then compared to the known addresses involved with anomalous behavior to analyze for classification and f-score accuracy. These values are determined as:

$$F-Score = \frac{TP}{\frac{1}{2}(FP+FN)}$$

$$Classification = \frac{TP+FP}{TP+TN+FP+FN}$$



The dotted red line represents the 4th degree polynomial trend line that is used to differentiate global and contextual outliers. All outliers marked with a black cross are labeled as suspicious and all suspicious marks that fall above the red line are labeled as anomalous.

Conclusions and Future Work

What the results of this study indicate is a reaffirmation that the ego-net features can be used to accurately classify nodes in a graph. The results also indicate that although the power law holds for a real graph even when broken down into multiple sub-graphs, there is no correlation between the power law exponent from one graph to the other. These results are validated with multiple tests at various polynomial degrees to assure the linear regression mapping wasn't polluting the findings. The f-score is the measurement that determines the algorithm's ability to recognize a distinct node as anomalous or not so the lack of significant change indicates that the algorithm did not increase in accuracy. The increase in classification score was due to the trend line comparison reducing the amount of false positives. The reason this didn't increase the f-score was because it also reduced the amount of true-positives the algorithm found. The general trend line merely truncated the list of detected nodes and didn't have an affect on predictive outcome.

References

[1] L. Akoglu, M. McGlohon, and C. Faloutsos,
"Oddball: Spotting Anomalies in Weighted Graphs". [Online]. Available:
http://http://www.cs.cmu.edu/~mmcgloho/pubs/pakdd10.pdf
[Accessed June 29, 2022]

[2] R. Fontugne, P. Borgnat, P. Abry, K. Fukuda.
"MAWILab: Combining diverse anomaly detectors for automated labeling and performance benchmarking". ACM CoNEXT 2010. Philadelphia, PA. December 2010. [Online]. Available:
http://conferences.sigcomm.org/co-next/2010/CoNEXT_papers/08-Fontugne.pdf [Accessed June 29, 2022]

[3] Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python,"
Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011. [Online]. Available:http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

[4] J. Dean, S. Ghemawat, "MapReduce: simplified data processing on large clusters".
Communications of the ACM, vol. 51, issue 1, pp. 107-113, January 2008. [Online]. Available: https://doi.org/10.1145/1327452.1327492

[5] Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart,
"Exploring network structure, dynamics, and functions using NetworkX" *Proceedings of the 7th Python in Science Conference (SciPy2008)*, Gael Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11-15, Aug. 2008.

[6] *mrjob* (2015), https://mrjob.readthedocs.io/en/latest/ [Online]

[7] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357-362 (2020). DOI: 10.1038/s41586-020-2649-2. [Online}. Available:https://www.nature.com/articles/s41586-020-2649-2

[8] V. Rossum, Guido and Drake Jr, Fred L. Python reference manual. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

Acknowledgements

The work is funded by NSF CNS-2149591 under Research Experiences for Undergraduates (REU) Program.