Master's thesis
Master's Programme in Data Science

Evaluation of Node Classification Methods in Citation Networks

Samu Suomela

December 6, 2021

Supervisor(s): Michael Mathioudakis

Examiner(s): Michael Mathioudakis
Jyrki Kivinen

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Master's Programme in Data Science

Tekijä — Författare — Author

Samu Suomela

Työn nimi — Arbetets titel — Title

Evaluation of Node Classification Methods in Citation Networks

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	December 6, 2021	20

Tiivistelmä — Referat — Abstract

Large graphs often have labels only for a subset of nodes. Node classification is a semi-supervised learning task where unlabeled nodes are assigned labels utilizing the known information of the graph. In this thesis, three node classification methods are evaluated based on two metrics: computational speed and node classification accuracy. The three methods that are evaluated are label propagation, harmonic functions with Gaussian fields, and Graph Convolutional Neural Network (GCNN). Each method is tested on five citation networks of different sizes extracted from a large scientific publication graph, MAG240M-LSC. For each graph, the task is to predict the subject areas of scientific publications, e.g., cs.LG (Machine Learning). The motivation of the experiments is to give insight on whether the methods would be suitable for automatic labeling of scientific publications.

The results show that label propagation and harmonic functions with Gaussian fields reach mediocre accuracy in the node classification task, while GCNN had a low accuracy. Label propagation was computationally slow compared to the other methods, whereas harmonic functions were exceptionally fast. Training of the GCNN took a long time compared to harmonic functions, but computational speed was acceptable. However, none of the methods reached a high enough classification accuracy to be utilized in automatic labeling of scientific publications.

ACM Computing Classification System (CCS):
Computing methodologies → **Machine learning** → Machine learning approaches → **Neural networks**
Computing methodologies → **Machine learning** → Learning settings → *Semi-supervised learning settings*

Avainsanat — Nyckelord — Keywords

node classification, machine learning, graph convolutional neural networks

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1. Introduction

The number of scientific publications is increasing at an exponential rate [15]. The sheer volume of publications is proving to be challenging for humans to manually keep up with and errors are imminent [16]. Thus, the need for automation has increased significantly. Real-life examples of this problem already exist: *U.S News and World Report* provided rankings on research universities based on incorrect data that may have led to students choosing their universities with poor information and funding being allocated incorrectly [1, 16]. In this thesis, three different node classification methods are evaluated in the task of labeling scientific publications.

The journal impact factor (JIF) is a common metric used to measure the quality of a journal. It is calculated by aggregating the number of citations to each article published in a journal and then dividing by the number of articles that are possible to cite [14]. Unfortunately, the JIF can be manipulated, and such manipulation is not uncommon. For example, scholars may have experienced coercive citation, where editors recommend authors to cite the editor's journals to pad their own impact factor [6].

Though many scientific articles are still being published in journals, the importance of online services as an outlet for new research has increased in recent years [15]. *Microsoft Academic Services* (MAS) considers this change in scholarly communications and takes new outlets, such as web articles, into account [15]. Technological advancements enable better acquisition of data and create higher quality data because human error can be left out. Therefore, mistakes caused by data such as the previously mentioned university rankings should be less frequent. *Microsoft Academic Graph* (MAG) is the data component of MAS and it is the representation of scholarly communications acquired by the technology of MAS [15].

Open Graph Benchmark (OGB) provides large graph datasets that are aimed to help researchers advance in graph machine learning [8]. This thesis will focus on MAG240M-LSC dataset that is a knowledge graph of scientific publications, authors and their affiliations extracted from MAG. The full graph is large, with a total of 240 million nodes and 1.7 billion edges. Approximately 120 million nodes represent scientific publications and only 1.4 million of those nodes have labels, which describe

153 different subject areas of the publications. Subject areas describe the primary topic of the publication, e.g., cs.LG (Machine Learning). The task is to predict the labels of the unlabeled publication nodes. Currently, the subject areas are manually labeled by authors and arXiv moderators. Automatic subject area assignment would significantly decrease the labeling effort of human individuals in the future. Additionally, existing non-arXiv papers could be classified accurately, improving the search and organization of academic papers.

The task of labeling nodes in a network that can be represented as a graph is a *node classification problem* [2]. The setting can be viewed as a semi-supervised learning task, where unlabeled nodes are assigned labels based on the existing label information. Node classification methods aim to provide accurate labels for the nodes efficiently. The aim of this thesis is to test three different node classification methods and evaluate their accuracy and efficiency with five different sized subgraphs of the MAG240M-LSC graph. The tests provide insight into their suitability for automatic labeling of scientific publications. The methods tested in this thesis are label propagation, harmonic functions with Gaussian fields and Graph Convolutional Neural Network. The results in Chapter 5 show that label propagation algorithm has the highest node classification accuracy, yet it is significantly slower than the other two methods. On the other hand, the harmonic function algorithm is extremely fast compared to the other methods, but the algorithm's node classification accuracy is not sufficiently high for automatic labeling of the MAG240M-LSC graph. GCNN scales well with the size of the graphs but has a substantially lower node classification accuracy than the other two methods.

The rest of the thesis is structured as follows: in Chapter 2, related literature will be discussed, and the theoretical background of the three node classification methods will be introduced. After that, in Chapter 3, the MAG240M-LSC graph will be introduced in detail, along with details on the creation of subgraphs for the experiments. Then, in Chapter 4 the three methods will be discussed in the context of the experiments. In Chapter 5, the results of the experiments will be discussed. Finally, in Chapter 6, final thoughts of the results are discussed, along with suggestions for future work.

2. Key Concepts and Related Literature

To introduce the concept of a graph, let us consider a network of scientific publications. Let V be the set of nodes that represent the scientific publications, e.g., articles and papers. Let E be the set of edges that represent citations from one publication to another. An edge (i, j) indicates that there is a citation between nodes i and j. A graph $G(V, E)$ represents the full citation network.

Let L be the set of labels for each publication. For example, labels can give information about the subject area of the publication. However, labels are often given by the authors and/or the publisher. Thus, labels might be impartial, incomplete, or incorrect [2]. For example, supervised learning tasks require labels for training. To tackle the issue, methods that assign labels for unlabeled data points are essential. In this chapter, node classification and related methods are discussed.

2.1 Node Classification

The surge of popularity in social networks has led to an abundance of information on individuals [2]. The notion of similarity between nodes is important. This similarity proves to be useful for applications such as recommendation systems.

Two key concepts identified by social sciences are important: *homophily* and *co-citation regularity* [2]. Homophily in the context of graphs means that a link between nodes correlates with the similarity between the nodes. For example, on Facebook, friends are often similar in age and location. Co-citation regularity is quite close to homophily: if two nodes share common features, they are likely to be similar in other ways too. For example, in Spotify, if two people listen to the same music, they both are likely to enjoy the same new music.

Applications of graph-structured data can be seen everywhere. Recommendation systems, such as Spotify's music recommendations and Netflix's movie recommendations utilize the information of the user network. As another example, Facebook and LinkedIn predict friendships and connections based on their user graphs.

Node classification is a task where previously unlabeled data points are given labels. Often with real world datasets, only a part of data is labeled. This can be caused by many reasons. For example, users may choose not to add certain information, or the information may simply be outdated [2]. This lack of information leads to the "node classification problem": how are we able to provide accurate labels to data points that are unlabeled? There are different angles that can be taken to distinguish between methods that classify nodes. *Iterative methods* use graph information as features, and methods that use *random walks* to propagate existing labels to unlabeled points [2].

2.2 Label Propagation

Label propagation draws its inspiration from methods like *k-nearest neighbors* [19]. In the algorithm, unlabeled nodes receive labels from their neighbors based on distance. Thus, labels push through unlabeled data. In case of a tie, one of the majority labels is chosen, for example, randomly [4]. The algorithm stops once it converges [19]. It can also be manually stopped by the user after a certain number of iterations.

Label propagation is a useful algorithm as it requires no knowledge of the network structure [19]. However, it may not perform as well if the distribution of labels is uneven [17]. For example, if one label is much more common than others, it is likely that unlabeled nodes are assigned that label instead of a more uncommon one. If the distribution of known labels does not represent the real distribution of all labels, the algorithm will be biased.

2.3 Harmonic Functions with Gaussian Fields

One way of labeling unlabeled nodes is viewing the graph in terms of a Gaussian random field. Let us assume that a connected graph $G(V, E)$ with an $n \times n$ symmetric weight matrix W on the edges is given for n data points. For example, the weight matrix can be

$$w_{ij} = -\exp \frac{(x_i - x_j)^2}{\sigma^2}$$

where x_i and x_j are the ith and jth nodes in the graph, σ is the length scale hyperparameter and w_{ij} denotes the weight of the edge connecting node i and j. Thus, in Euclidean space, nodes that are close to each other are given larger weight [20].

Then, function $f : V \to \mathbb{R}$ is computed and the values of f are used to assign values for unlabeled nodes. The probability distribution on functions f is discovered by forming a Gaussian field, which normalizes all functions f on the labeled data. Each

unlabeled data point has a value equal to the average of f at neighboring points due to the *harmonic* property of f [20].

Using Gaussian fields changes the sample space from discrete into continuous. Therefore, the most probable field is unique and characterized by harmonic functions [20]. Harmonic functions with Gaussian fields follow the paradigm of nearest neighbor classification methods closely, as new labels are computed based on random walks on the graph.

2.4 Neural Networks

Deep Learning has allowed massive progress in different fields of machine learning. For example, in 2015, *AlphaGo* managed to beat the European Champion in a full-sized game of Go [13]. This achievement was thought to be years away at the time. Similarly, other fields have applied neural networks to achieve better results in existing research problems.

Computer vision is another field where Deep Learning has improved results significantly. Convolutional Neural Networks have allowed Deep Learning methods to surpass traditional computer vision methods in multiple areas, e.g., image classification and object detection [11].

Many existing node classification methods assume that connected nodes share a label and that the existence of an edge indicates similarity [9]. Often these methods only propagate label information from the neighbors of the nodes, without considering feature information if that is available. Node features could provide additional information on the similarities between nodes, improving accuracy. Consequently, classification becomes computationally more expensive.

Neural networks have been applied to graphs earlier. Graph data was often preprocessed into a vector representation to allow analysis to be performed [7]. However, the preprocessing might lead to a loss of significant information and results depend on the preprocessing stage. To avoid the loss of information, the first Graph Neural Network operated directly on the graph data [7]. However, this method was not feasible for large datasets [18]. Since then, like in Computer Vision, the idea of adding a convolutional layer to neural networks prompted many new, successful methods for analyzing graph-structured data [18]. The approach of Graph Convolutional Neural Networks was initiated in 2013, when graph convolutions were based on spectral graph theory and experiments gave promising results [3, 18]. After that, improvements have been made in many areas of graph analysis, including node classification and link prediction. For example, GCNN achieved better classification accuracy and computational efficiency in the popular Citeseer, Cora and Pubmed citation datasets [9].

Different GCNN's have achieved success in multiple domains. However, they often require storing the full graph Laplacian in memory during training. Thus, they are limited by graph size [18]. PinSAGE utilized random walks to sample the node neighborhood to avoid constructing the full graph Laplacian to scale the method to billions of nodes and edges. Additionally, PinSAGE utilized MapReduce [5] pipeline to allow efficient training.

3. Dataset

Universities are mostly ranked by commercial entities [1]. These rankings can be inherently flawed due to many reasons. For instance, *U.S News and World Report* ranks Ph.D programs in computer science without proper understanding to the field [1]. Inaccurate rankings seem to be caused by incomplete data rather than the analysis performed on the data [15]. Significant scientific outlets have been left out from the data. For example, *Journal of Machine Learning Research* is a particularly relevant source of scientific articles in the field of computer science that is not included in the data used to report the ranking of *U.S News and World Report* [15].

Scientific discussion has gone through a notable change in the past few years. For instance, transparent online discussion and data sharing has increased [16]. At the same time, number of scientific publications in journals keeps on growing at an exponential rate [15]. Along with technological advancements in artificial intelligence (AI), Microsoft Academic Services (MAS) is a project that attempts to capture a knowledge graph of scientific publications with as much detail as possible [15]. Microsoft Academic Graph (MAG) is the data component of MAS that is acquired by AI automatically and is kept up to date with web scraping [15].

3.1 Microsoft Academic Graph

Microsoft Academic Graph (MAG) knowledge graph is a heterogeneous graph that consists of different scholarly entities and their relationships [15]. The scientific publications are in the center of the graph. There are three types of nodes: publication nodes, author nodes and institution nodes. If a directed edge from a publication node to another publication node exists, it indicates a citation from the starting node to the end node. Similarly, a directed edge from a publication node to an author node indicates authorship of the publication. Lastly, a directed edge from an author node to an institution node indicates an affiliation between the two.

3.2 Open Graph Benchmark

Most used graph datasets are inherently too small compared to graphs found in real life applications. For example, for node classification, many models are developed and tested using Cora, Citeseer and Pubmed datasets, which only include 2700 to 20000 nodes [8]. Due to the relatively small size of these datasets, some models are not scalable in larger graph environments [8]. Additionally, Cora and Citeseer have quality issues. For instance, in the Citeseer dataset, 61.8% of node features leak label information, and 4.8% of nodes are duplicates [21].

Open Graph Benchmark (OGB) provides graph datasets that are large enough that models developed using their data can be applicable to real-world graphs, i.e., 1 million nodes. Additionally, these datasets provide various tasks in various domains, such as node classification, link prediction and graph prediction [8].

3.3 MAG240M-LSC Graph

MAG240M-LSC is a heterogeneous graph extracted from MAG. It is a one of the OGB datasets introduced in Section 3.2. In total, the graph includes 122 million publication nodes, 122 million author nodes and 26 thousand institution nodes. Between the nodes, there are 1.3 billion citation edges between the publication nodes, 386 million edges from author nodes to publication nodes and 45 million affiliation edges from author nodes to institution nodes. The graph diagram is shown in Figure 3.1. The MAG240M-LSC dataset can be downloaded and prepared using OGB Python package.[1] Additionally, the model evaluation and test results can also be handled with the same package.

Of the 122 million publication nodes in MAG240M-LSC, 1.4 million are manually labeled arXiv papers, divided into 153 arXiv subject area classes, e.g., cs.LG (Machine Learning). All publication nodes are numbered from 1 to n, where n is the total number of publication nodes. The rest of the publication nodes are unlabeled but are included in the graph to provide structural information. The label information is represented in a vector, where value of index $[i]$ is the label of node i. The value integer between 1 and 153, representing the different subject areas, if the label is known.

Each publication node in the raw data includes its concatenated title and abstract that have been passed to a RoBERTa sentence encoder [10, 12], returning a 768-dimensional feature vector for each paper. The resulting feature matrix is extremely large, approximately 175GB.

[1] https://github.com/snap-stanford/ogb

The edgelist is a numpy array of shape (2, num_edges), where [0,i] is the source node of i-th edge, and [1,i] is the target node of i-th edge. Each edgetype can be accessed through edge index.

MAG240M-LSC is used in the KDD Cup 2021 competition.[1]

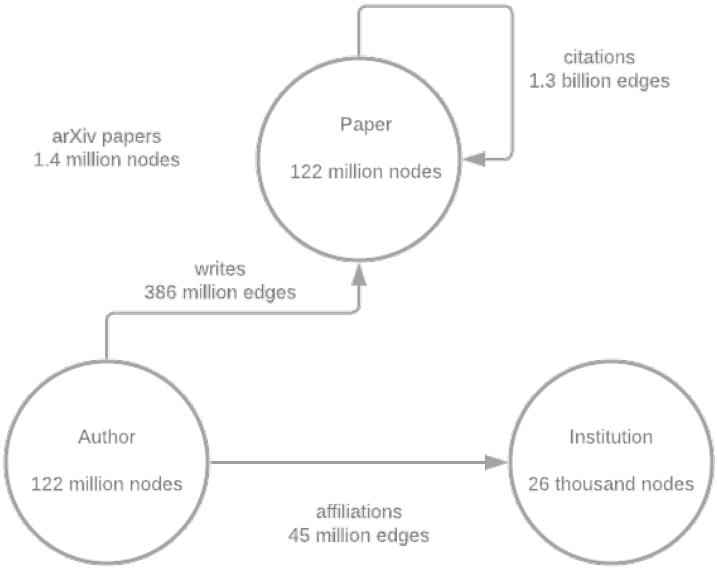

Figure 3.1: Diagram of the MAG240M dataset, drawn on Lucidchart

3.3.1 MAG240M-LSC Data for the Experiments

For the experiments, smaller subgraphs of the MAG240M-LSC graph are created. First, a root paper that has a label is given as an input to the algorithm. In this thesis, the root paper was chosen in a way that will lead to graphs that have a large total number of labels. The algorithm selects the neighbors of the root node and adds the nodes and edges to the graph, creating a connected subgraph of the full graph. Then, the algorithm selects the neighbors of each new node, and adds the nodes and edges to the graph. This process is repeated seven times in total. For the last five iterations of the algorithm, the graph is saved as a numpy array, along with features and labels of the nodes in the graph. Thus, the algorithm outputs five connected graphs of different sizes as well as the feature and label information. Algorithm 1 shows the pseudocode of the subgraph creation.

[1] https://ogb.stanford.edu/kddcup2021/

Algorithm 1 Subgraph creation

Input: Root node

Output: Five connected graphs as numpy arrays

$i = 0$
Select root node
for $size = 1, 2, \ldots, 7$ **do**
    Select neighbors of nodes in current graph that are not yet in the graph
    Retrieve the edges, features, and labels for the nodes in the graph
    if $i = 2$ **then**
        Save nodes, edges, features, and labels to a file
    else $i = i + 1$
    end if
end for

4. Methodology

Experiments are done on five graphs retrieved from the MAG240M-LSC data. Each graph is different in size to examine how well each method scales. Table 4.1 shows the statistics of each graph.

Three different node classification methods are tested on each of the five graphs. The objective for each method is to predict the label of the nodes in the graph. The labels are integers that represent different subject areas of the scientific publications. In the experiments, the node classification methods are tasked with predicting the subject areas for each node in the graph.

All experiments were done using computational resources provided by the Finnish Grid and Cloud Infrastructure (FGCI). For each experiment, 32GB of CPU-memory was requested from the system. If the system is not busy, more resources may be allocated. All experiments were given 24 hours to conclude. All code used for the experiment are public and available in GitHub.[2]

Size	Nodes	Edges	Labeled Nodes	Unique Labels
Small	1582	3092	233	16
Medium	7841	19366	793	21
Large	31189	90009	2320	32
XL	94690	353231	5855	45
XXL	256092	1016389	11852	63

Table 4.1: Statistics of different graphs used in the experiments

4.1 Label Propagation Algorithm

The assumption of nodes appearing near each other indicating similarity seems plausible for a citation network such as this data. It is likely that papers cite other papers

[2] https://github.com/samu-suomela/MAG240M_Thesis

within their field. Even if there are citations to papers in other fields, they should be outnumbered by citations to the same topic.

For the experiments, label propagation is performed as follows. After 20% of known labels are hidden, the algorithm is given the graph as input. Then, the node list is looped through. The nodes that have a label at the beginning are ignored, as those labels are known to be true. For each node, we loop through its neighbors and select the most frequent label among them and assign that label to the current node. If none of the neighbors have labels, we ignore that node and move to the next one. Once each node has been assigned a label and they no longer change, the algorithm has converged, and label propagation is complete. The algorithm outputs the propagated labels as a matrix. The hidden 20% of the labels are then compared to the propagated labels to evaluate accuracy. Algorithm 2 shows the pseudocode for label propagation.

Algorithm 2 Label propagation

```
while converged do
    converged = True
    for node = 1, 2, ... do
        Select most frequent label among neighbors
        if no label then
            continue
        else
            Assign most frequent label for current node
            converged = False
        end if
    end for
    if converged then
        break
```

4.2 Harmonic Function Algorithm

Classifying nodes with the Harmonic Function algorithm is straightforward. *NetworkX* is a Python library that can be utilized for graph analysis. Node classification using harmonic functions can be done utilizing NetworkX's ready-made function.[1] First, a NetworkX Graph-object is constructed from the edge data. After removing 20% of the labels, each known label is assigned to its corresponding node. Then, the algorithm is

[1] https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.node_classification.hmn.harmonic_function.html

given the Graph-object as input and the algorithm outputs a list of predicted labels for all nodes. The function is capped at a maximum of 30 iterations.

4.3 Graph Convolutional Neural Networks

As mentioned in Chapter 2, Deep Learning has improved different graph analysis tasks by a significant margin. For the experiments, a neural network that uses first-order approximations of spectral graph convolutions as the convolutional architecture is used [9]. The method was originally tested on three different citation networks. Thus, the method should perform great in this experiment. The code used in the experiments is available in GitHub.[1]

To utilize the GCNN, significant data preprocessing is required. First, a label matrix with n rows and m columns is created, where n is the number of nodes in the data, and m is the number of unique labels. Each row has a 0 in each column if the node corresponding to that row does not have a label. If the node has a label, it has the value 1 in the column that is the same number as the label, and 0 elsewhere. All rows with labels are then split into training and testing matrices, which are both saved as files to be used with GCNN. Additionally, a label matrix without the test labels is saved.

Similarly, features of each paper are saved. Three separate feature matrices are created, one with only the nodes that have labels that are used for training, one with only the nodes that have labels that are used for testing, and a feature matrix including the unlabeled nodes, but excluding the test features. These three matrices are then saved as sparse matrices to save memory and speed up the processing.

Finally, the graph itself is saved as a dictionary. Each key in the dictionary is a node, and the value is a list of the indices of the neighboring nodes.

GCNN has adjustable hyperparameters. For the experiments, the neural network had 16 units in the first hidden layer. The model was trained for 200 epochs with an initial learning rate of 0.01 and dropout rate of 0.5. Modifying the hyperparameters can have an impact on the neural network's performance, but brief experiments with the smallest graph led to the choice of the reported hyperparameters.

[1] https://github.com/tkipf/gcn

5. Results

The three methods described in Chapter 4 were tasked to predict the labels of nodes on five datasets of different sizes, each extracted from the MAG240M-LSC citation graph. Each label describes the subject area of the publication node. For each method, 80% of the labels were used for training, while the remaining 20% were used for evaluation.

Label propagation algorithm reached the highest classification accuracy among the three methods. For the two smallest graphs, the classification accuracy was slightly below 40%, while for the third largest graph the accuracy was above 60%. This might be caused by uneven distribution of the labels, where many of the unlabeled nodes are given the most frequent label. However, for the two largest graphs, the algorithm did not converge within 24 hours. It is possible that if the algorithm had been given enough time to converge, node classification for the two largest graphs would have provided accurate labels. However, the low computational speed makes the method impractical.

Node classification accuracy for the harmonic function algorithm was between 20% and 40% across all five graphs. The low prediction accuracy indicates that harmonic functions are not suitable for automatically predicting the subject areas of scientific publications.

While GCNN should perform well in a citation network, the node classification accuracy was unexpectedly low for each of the graphs. Even though GCNN was fed feature information about the publication nodes, the classification accuracy was only between 2% and 4%. Node classification results for all three methods are shown in Figure 5.1.

Label propagation algorithm did not scale well computationally as the size of the graphs increased. The algorithm took over 3 hours to converge for the third largest graph and did not converge within 24 hours for the two largest graphs. Thus, label propagation, if not carefully optimized, is not suitable for automatic classification of scientific publications.

The harmonic function algorithm was fast across all graphs. Even for the largest graph, the algorithm took less than 2 minutes to classify the nodes, which is significantly faster than the other two methods. Considering both node classification accuracy and computational speed, harmonic function algorithm performed the best among the three

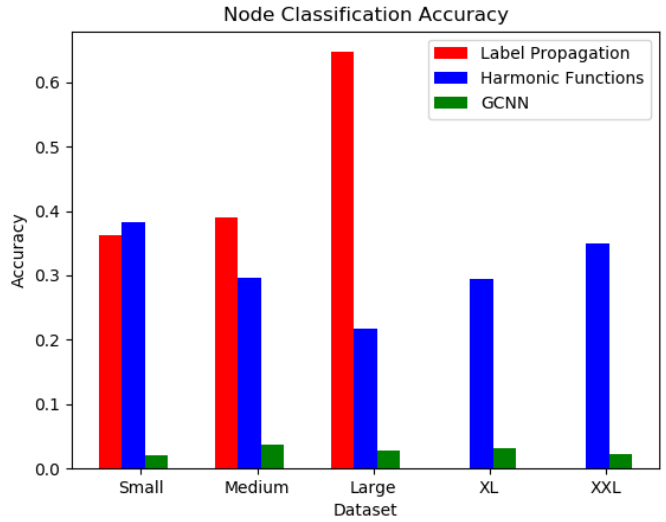

Figure 5.1: Node classification accuracy of each method

methods.

Classifying the nodes with GCNN took longer than with harmonic functions, though that is to be expected with the additional feature information. Computational speed of GCNN appears to scale well with increasing sizes of the graphs. Finding the right network architecture for automatic labeling might lead to better node classification accuracy. Computational speed for each of the three methods are shown in Figure 5.2.

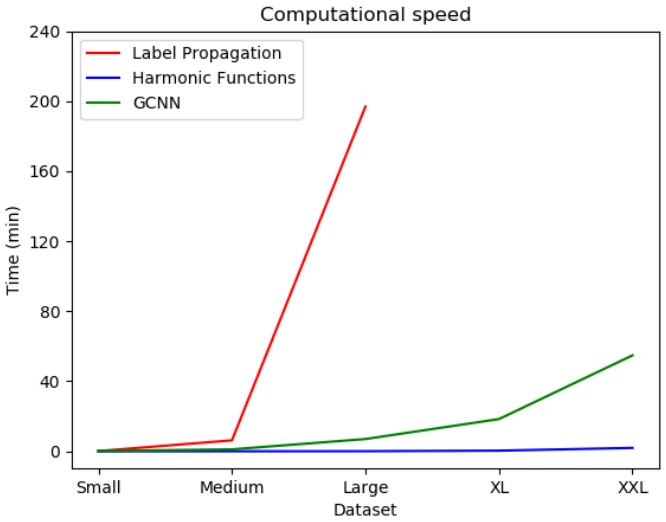

Figure 5.2: Computational speed of each method

6. Conclusion

In this thesis, three node classification methods were discussed: label propagation, harmonic functions with Gaussian fields and Graph Convolutional Neural Network (GCNN). Each method was tested on five different sized graphs, which were subgraphs of the massive MAG240M-LSC graph. The methods were compared between each other with regards to node classification accuracy and computational speed.

Harmonic functions and label propagation algorithms achieved mediocre results in node classification, while GCNN had performed poorly. On the other hand, label propagation scaled very poorly on larger datasets, while harmonic functions were incredibly fast across all graphs. GCNN was slower than harmonic functions but scales well with the size of the graph. Further investigation of GCNN's hyperparameters could improve the node classification accuracy of the method. However, none of the methods had a suitably high node classification accuracy to be used in automatic labeling of scientific publications.

This work can be extended to many directions. For example, the sizes of the graphs in the experiments were only a comparably small subset of the full graph. Experiments using the entire graph could provide meaningful insight into the labeling task. For example, GCNN could benefit from an even larger input.

MAG240M-LSC contains plenty of more information that was not utilized in this thesis. For instance, author nodes could provide additional information on the graph structure, potentially increasing node classification accuracy.

Current ready-made implementations of label propagation were unsuited for this task. For example, NetworkX's label propagation algorithm does not use any prior knowledge of labels and detects communities based on structure alone. An efficient implementation of the label propagation algorithm would benefit graph analysis, particularly with large graphs.

Acknowledgements

The authors wish to thank the Finnish Grid and Cloud Infrastructure (FGCI) for supporting this project with computational and data storage resources.

Bibliography

[1] E. Berger, S. M. Blackburn, C. Brodley, H. V. Jagadish, K. S. McKinley, M. A. Nascimento, M. Shin, K. Wang, and L. Xie. Goto rankings considered helpful. *Commun. ACM*, 62(7):29–30, 2019.

[2] S. Bhagat, G. Cormode, and S. Muthukrishnan. Node classification in social networks. *Social Network Data Analytics*, pages 115–148, 2011.

[3] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs, 2014.

[4] G. Cordasco and L. Gargano. Community detection via semi-synchronous label propagation algorithms. pages 1–8, 2011.

[5] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, pages 137–150, 2004.

[6] E. Fong and A. Wilwhite. Authorship and citation manipulation in academic research. *PLoS ONE*, 12(12), 2017.

[7] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2, 2005.

[8] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs, 2021.

[9] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks, 2017.

[10] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[11] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. Velasco-Hernandez, L. Krpalkova, D. Riordan, and J. Walsh. Advances in computer vision. *Advances in Intelligent Systems and Computing*, 2020.

[12] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2017.

[13] D. Silver, A. Huang, and C. Maddison. Mastering the game of go with deep neural networks and tree search. 2016.

[14] J. P. Tennant, H. Crane, T. Crick, J. Davila, A. Enkhbayar, J. Havemann, B. Kramer, R. Martin, P. Masuzzo, A. Nobes, C. Rice, B. Rivera-López, T. Ross-Hellauer, S. Sattler, P. D. Thacker, and M. Vanholsbeeck. Ten hot topics around scholarly publishing. *Publications*, 7(2), 2019.

[15] K. Wang, Z. Shen, C. Huang, C.-H. Wu, Y. Dong, and A. Kanakia. Microsoft Academic Graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413, 02 2020.

[16] K. Wang, Z. Shen, C. Huang, C.-H. Wu, D. Eide, Y. Dong, J. Qian, A. Kanakia, A. Chen, and R. Rogahn. A review of microsoft academic services for science of science studies. *Frontiers in Big Data*, 2:45, 2019.

[17] Y. Yamaguchi and K. Hayashi. When does label propagation fail? a view from a network generative model. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3224–3230, 2017.

[18] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, 2018.

[19] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. 2003.

[20] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pages 912–919. AAAI Press, 2003.

[21] X. Zou, Q. Jia, J. Zhang, C. Zhou, Z. Yao, H. Yang, and J. Tang. Dimensional reweighting graph convolution networks, 2020.