**COMPARATIVE CASE STUDY BETWEEN GATED GRAPH NEURAL NETWORKS**

**VERSUS**

**RELATIONAL GRAPH CONVOLUTIONAL NETWORKS FOR THE VARIABLE MISUSE TASK**

**USING PYTHON AND DEEP LEARNING**

**By**

**Mukund Raghav Sharma**

A Capstone Project Paper Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in

Data Science

University of Wisconsin – La Crosse

La Crosse, Wisconsin

May 2020

**DEDICATION**

**To my grandparents and my favorite planet, Saturn.**

**ABSTRACT**

**GATED GRAPH NEURAL NETWORKS VS RELATIONAL GRAPH CONVOLUTIONAL NETWORKS FOR THE VARIABLE MISUSE TASK**

**MUKUND RAGHAV SHARMA**

Identifying bugs in source code has been an extremely important part of software development since the inception of the industry. The majority of static analysis, the analysis of software without actually executing programs, is rule based without much involvement of deep learning until fairly recently. This paper engages in a comparative study of determining the more performant graph neural network model on the basis of test accuracy between Gated Graph Neural Network (GGNN) models and Relational Graph Convolutional Network (RGCN) models on the Variable Misuse Task, a prediction task involving choosing the correct variable based on all the variables of the same type in a particular scope.

The data is of source code from the files of 24 trending C# repositories that are converted into a modified Abstract Syntax Tree to represent a directed graph whose vertices that represent the tokens and relationships between the tokens are represented by edges. Each of these vertices are associated with one of the aforementioned type of networks for the training phase after a particular embedding is computed for each token.

The comparison to decide the more efficient model is based on the test accuracy of all the repositories, an esoteric repository and an extremely popular repository to cover the spectrum of different types of repositories. The results show that the RGCN based models outperformed the GGNN models for all cases, albeit, within a < 5% range.

*Keywords: Deep Learning, Graph Neural Networks, Tensorflow, Sequence Models, Convolutional Models, Learning from Code, Static Analysis.*

**TABLE OF CONTENTS**

Abstract

Introduction

Literature Review

Project Overview

Graph Neural Networks 101

Preliminary Steps

Data Exploration

Methodology and Outcomes

Results

Conclusion

**CHAPTER 1**

**INTRODUCTION**

**Project Background**

Since the inception of the computing industry more than 70 years ago, the need for correct and efficient verification tools has been extremely desirous in the software engineering world. Static Analysis tools or tools that analyze the code without executing the program boost the productivity of the developer by highlighting some bugs at development time rather than during when the code is running in production.

Deep learning application in the field of learning from source code and static analysis is still in its nascent phase and a large part of the current work doesn’t take advantage of the representational power of both the syntactic and semantic nature of the code. For example, shallow representations of source code is prevalent in recent research such a simple sequence of tokens such as by work done by Hindle et al. (2012) [1] or flat dependency networks of variables by Raychev et al. (2015) [2].

Recent work from Microsoft Research involves inculcating both the semantic and syntactic nature of source code by representing programs as directed graphs [3] and applying deep learning via sequential or convolutional neural networks to create graph neural network models. Using graph neural network models has proved to more easily solve state-of-the-art problems in the space of learning from code as the representational capacity of these models reduces the need for a large training set as well as meticulously encapsulates the semantic relationships between variables and types of the source code.

In this paper, I use the work by Microsoft Research in the field of Graph Neural Networks to conduct a comparative study between two different graph neural networks namely, Gated Graph Neural Networks (GGNN) and Relational Graph Convolutional Networks (RGCN), on the Variable Misuse Task to discern the better performing model on the basis of test accuracy. The Variable Misuse Task is a prediction based task on source code involving predicting the variable that most accuracy fits the current context from all the variables of the same type in a particular scope.

The Variable Misuse Task is a simple yet an extremely important task that has its uses in the world of static analysis. Microsoft Research’s application of Graph Neural Networks on the Variable Misuse task has caught bugs that had been deployed in production for important repositories such as RavenDB and Rosyln [3]. Further application of Graph Neural Networks on newer tasks show great promise and have already started changing the way code is tested and validated.

The input data used for this model is that of the top 24 trending C# repositories on Github. There are three main experiments conducted to discern the better performing model on the basis of test accuracy and these are training and testing on the source code of all the repositories, an esoteric repository and an extremely popular repository. The rationale here was to gain insight about how the GGNN and RGCN models would fare in different cases and which one of the two would prove to be the better predictor.

**Objectives**

The main objective of this project is to compute the test accuracies for the GGNN and RGCN models for three experiments namely, training and testing on data from 27 of the most popular C# repositories on Github, an esoteric repository and a popular repository, in an effort to discern the more performant model.

This process of computing the test accuracies involved enumerating through some minor, more granular, objectives and these are:

1. Obtaining, Cleaning and Understanding the Input Data.
2. Choosing, from the trending C# repositories, which one would characterize as the esoteric and popular one to conduct our experiments on.
3. Establishing a working pipeline through which the experiments will be conducted.
4. Interpreting results i.e. test accuracy of the experiments to land on a conclusion.

**Rationale and Inspiration for undertaking the project**

The constant strive to build better development tools that will improve the overall user experience is one I am starting to familiarize myself with more since I joined Microsoft’s Developer Division as a software engineer where I work on the Performance and Reliability of Visual Studio. My role involves creating processes that improve the prevention of performance regressions by early detection before they reach the end user. My experience with performance engineering and testing on a large code base coupled with my strong inclination towards deep learning got me going down a path to choose a topic for my capstone that would fit the said intersection.

After discovering Microsoft Research’s ground breaking work, I was convinced I needed to know the ins and outs of the details of how they improved state-of-the-art research models. And as a result to add more specificity to my topic, I decided to choose between a sequence based model (GGNN) and one convolutional based model (RGCN) on a the Variable Misuse Task. Also, I was lucky enough to have all my questions answered within a couple of hours by the good people at Microsoft Research and this was another inspiration booster i.e. to work on ground breaking work with individuals who were extremely eager to help out in improving the understanding of their work.

The choice of repositories were amongst the top trending C# repositories that I have used professionally which made pursuing the data exploration all the more exciting as it provided me with a reason to dive deep into the code, as well. Conducting the comparison on repositories like Newtonsoft.Json was enticing as these are libraries I use on a daily basis and have used all my career being a .NET developer.

**CHAPTER 2**

**LITERATURE REVIEW**

**Previous Work**

Graph Neural Networks

2005

2016

Gated Graph Neural Networks

Graph Convolutional Networks

2017

Neural Message Passing

Programs As Graphs

2018

**Fig. 1: Timeline of Graph Neural Networks Research**

Researching through a lot of the related past work in the field of Graph Neural I was able to narrow down the timeline of key developments in this field details of which are as follows:

**2005**: Gori et al. proposed the notion of Graph Neural Networks that was the seminal research paper in this field that laid down the foundation of future work. Although, deep learning wasn’t at the forefront of statistical research when this paper was written at it is now, important fundamental concepts such as training of neural networks on graphs as well message passing emerged.

**2016**: Li et al. took the core concept of Graph based networks and applied sequence based concepts such as gated cells to neural networks to prevent loss of importance of information in large, extremely spread out sequences represented as graphs. The most important aspect of this model was the propagation model that took the messaging passing logic and applied recurrent gated cells at each vertex.

**2017**: Liao et al proposed idea of Neural Messaging Passing where information between the neighbors of vertices is shared and therefore, not only does each vertex in a graph, through training, have information about its initial state but also other vertices. I will be expounding on this a lot more in a later chapter as this concept has be explained thoroughly for full effect but to summarize, neural message passing allows graph neural network models to make use of their initial state but as well as the state of other vertices in an effort to increase overall representational capacity that requires fewer training data points.

**2017**: Kipf et al. introduced the concept of Graph Convolutional Networks that took plain vanilla graph neural networks and in state update training step used convolution of the state of the neighbors as the mechanism of message passing. The premise here is that the Convolution operation that is typically used in the case of computer vision can be applied in the case of graph neural networks in the case of message passing where each vertex can gain further insight about it’s position with respect to other vertices.

**2018:** Allamanis et al. applied concepts from previous research in the context of learning from programs following the simple yet powerful assertion that source code can be represented as graphs where the vertices represent the variables and it’s associated information and the edges represent the relationship between the variables. Additionally, taking advantage of the semantic and syntactic nature of source code adds more detail to the input of the graphs that can result in quicker training. The majority of this report is built on this paper and its applications and further details will be elucidated upon in future chapters.

The aforementioned concepts at this point might seem difficult to grasp as they aren’t fully elucidated on; I plan to cover the basics of all the concepts in one of the future chapters that’ll simplify the subject matter. Needless to say, the extensive research on this topic of graph neural networks is truly inspiring and exciting as well as growing exponentially with developments in multiple dimensions.

**CHAPTER 3**

**PROJECT OVERVIEW**

**General Approach / Layout**

1. The basic steps undertaken to approach this project include:
2. Defining the Experience, Task and Performance Metric in the style of the formal machine learning once the problem statement was identified that we’ll base the comparison on.
3. Acquisition and exploration of the input data source.
4. Deep diving into the source code associated with the Learning to Represent Graphs library to gain insight about how the code is executed.
5. Once the insight is gained to run the code, customize the code to accommodate for ease of use for my experiments by constructing a working pipeline that’s based on ease of use.
6. Defining the 3 experiments and which data will be used for them.
7. Training models with the training and validation input data on GGNN and RGCN models for the experiments.
8. Training with different hyperparameters that can be used to improve the overall validation accuracy.
9. Using the weights from the training process to compute the test accuracy for RGCN and GGNNs.
10. Aggregate results to discern which model was the more performant one.
11. Conclude with possible next steps.

Now that the goals and outline are set, I want to shed light on what’s actually happening under the hood during the training and evaluation of these models and define the task at hand with more detail.

**CHAPTER 5**

**GRAPH NEURAL NETWORKS 101**

**Goals**

In this section I plan to achieve the following goals:

1. Define and Describe the Variable Misuse Task in detail.
2. Explain how Graph Neural Networks work.
3. Deep Dive into Gated Graph Neural Networks.
4. Deep Dive into Relational Convolutional Neural Networks.
5. Define the Experience, Performance and Task for the problem.

By the end of this chapter, the reader should have gained some knowledge about graph neural networks and their details with relation to the comparative study between GGNNs and RGCNs.

**The Variable Misuse Task**

Properly defining and understand the task associated with our statistical learning algorithm is of paramount importance. The Variable Misuse Task, in a nutshell, is a task that requires the algorithm to predict the most fitting variable in a particular scope among all the variables that are of the same type. As an example, consider the following piece of code:

**Graph Neural Networks**

**Gated Graph Neural Networks**

**Relational Graph Neural Networks**

**Defining the Experience, Performance and Task**

Based on the task at hand

**CHAPTER 6**

**PRELIMINARY STEPS IN CASE STUDY**

**REFERENCES**

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009. The graph neural network model. IEEE Transactions on Neural Networks 20(1):61–80.

[1] Abram Hindle, Earl T Barr, Zhendong Su, Mark Gabel, and Premkumar Devanbu. On the naturalness of software. In International Conference on Software Engineering (ICSE), 2012.

[2] Benjamin Bichsel, Veselin Raychev, Petar Tsankov, and Martin Vechev. Statistical deobfuscation of android applications. In Conference on Computer and Communications Security (CCS), 2016.

[3] Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. 2017. Learning to Represent Programs with Graphs. arXiv:cs.LG/1711.00740