# *KCL_no UoL_letterhead_3#21DTEXT VS TREES VS GRAPHS:*

# *DEEP LEARNING TECHINIQUES FOR PROGRAM UNDERSTANDING*

## MSc Project Preliminary Report

Author: Olubusayo Akeredolu

Email : [olubusayo.akeredolu@kcl.ac.uk](mailto:olubusayo.akeredolu@kcl.ac.uk)

Student ID : 20107125

Supervisor : Dr. Maria Polukarov

April 1, 2022

CONTENTS

#### INTRODUCTION

##### PROJECT OVERVIEW

* What is Programming Language Understanding?
* Why am I interested in it/What makes it interesting?
  + The differences between NL and Source code:
    - NL is generally easier to understand.
    - NL requires less context. Give example of ‘The girl is in the class’ vs a source code example.
    - Source code contains rich structural and contextual information that cannot be found in NL e.g., control flow, conditional statements, functions, methods, inheritance in OOP, etc.
  + Computers not understanding programs
    - NLP models can’t tell the difference between regular NL and source code. They are treated the same, as I will demonstrate with the RNN model I build.

##### PROJECT AIMS & OBJECTIVES

1. Develop a method of converting Source code into either a directed graph containing nodes and edges.
2. Develop a method of converting Source Code into a tree structure.
3. Develop a Graph-based NN that classifies programs into one of 5 sorting algorithm categories.
4. Develop a Tree-based NN that classifies programs into one of 5 sorting algorithm categories.
5. Develop an RNN that classifies programs into one of 5 sorting algorithm categories.
6. Compare the performances and accuracies of each of the three models to determine which model is most suitable for carrying out classification tasks on source code.

##### PROJECT DOMAIN

* Artificial Neural Networks
* Graphs, Graph Theory & Graph Neural Networks
* Trees and Abstract Syntax Trees (ASTs)
* Machine Learning
* Deep Learning
* Deep Program Understanding
* Natural Language Processing

#### BACKGROUND & MOTIVATIONS

##### BACKGROUND

* Based on my interests in Programming Language Understanding.
* Applications of Graphs to programming languages: Every program can be represented as a flowchart/flow graph which is in turn a form of a directed graph.
  + Graphs carry a lot of information which can be utilised when trying to understand a program using NNs.
* Graph Neural Networks and how they work.
* Utilising ASTs to learn more about the underlying structure of a program.

##### RELATED WORK

* Talk about the relative newness of this field.
* Learning to Represent Programs with Graphs.
* TBCNN paper.
* GGSNN paper.

##### MOTIVATIONS

* Examining the different ways by which programming language understanding can be made possible.
* Investigating to see if computers can be taught (using NNs) to generate meaningful and usable code in the same ways that NLP models can generate meaningful and usable text.

##### --

#### SPECIFICATIONS, IMPLEMENTATION & SCHEDULE

##### IMPLEMENTATION OVERVIEW

* There are three different sections to this project.
  + The Text section: Building the RNN.
  + The Tree section: Building the Tree-Based NN.
  + The Graph section: Building the Graph-Based NN.
* Each section is further divided into three sub-phases:
  + The Implementation phase: where I implement the model and train it.
  + The Testing phase: where I test the model.
  + The Analysis and Evaluation phase: where I analyse and evaluate the results from the Testing phase.

##### DATASETS AND DATA COLLECTION

* The dataset is divided into two categories:
  + The Training data: which I use to train the models.
  + The Testing data: which I use when testing my models.
* In each category, there are five classes of sorting algorithms each:
  + Bubble Sort.
  + Insertion Sort.
  + Merge Sort.
  + Quick Sort.
  + Selection Sort.
* This data has been collected from various GitHub repositories.

##### TECHNICAL SPECIFICATIONS

* This project will be written using the following libraries:
  + The Python Standard Library.
  + TensorFlow.
  + Keras.
  + Sci-kit Learn.
* The Deep Learning techniques used will be supervised learning: classification based on the training dataset.
  + Back propagation
* I will build 3 separate NNs using TensorFlow, Keras and Sci-kit Learn.
* Descriptions:
  + RNN: Four layers: Input, output & two hidden layers. The number of neurons in each will be determined during the implementation process
  + Tree-Based NN: Input and Output layers. The number of hidden layers and the number of neurons in each will be determined during implementation
  + Graph-Based NN: Input and Output layers. The number of hidden layers and the number of neurons in each will be determined during implementation

##### PROGRAMMING LANGUAGES AND LIBRARIES

* Why Python?
* Why TensorFlow, Keras and Sci-kit Learn?
* Possibility of making use of more libraries outside of these three

##### IMPLEMENTATION SCHEDULE

* The implementation of this project has been divided into three sections as described in section 3.1. The order of implementation is as follows:
  + The RNN: This serves as a foundation to show the shortcomings of conventional NLP models when processing source code.
  + The Tree-Based NN: The implementation phase of this is broken down into two parts:
    - Creating the AST.
    - Running the model on the tree that has been generated.
  + The Graph-Based NN: This is also divided into the parts:
    - Generating the Graph based on the source code.
    - Running the model and training or testing it using the generated graph.

1. TESTING AND EVALUATION

##### TESTING

* Testing the model’s predictions using a dataset of 60 different sorting algorithms.
* Based on how efficient the model is in terms of time, there might be an increase in the number of python files used for both training and testing. This will provide a bigger and better picture of how the models really perform.

##### EVALUATION AND ANALYSIS OF RESULTS

* In my evaluation and analysis of the results, I will be focusing on two metrics:
  + Efficiency of the model: The speed at which it trains, and tests compared to the other two models. The less time taken to train, the more efficient the model is.
  + Classification Accuracy of the model: The level of accuracy of the model when making predictions on the testing data. The higher the accuracy, the more suited the model is for classifying sorting algorithms compared to the other models.

##### --

#### POTENTIAL APPLICATIONS AND POSSSIBLE FUTURE WORK

##### POSSIBLE FUTURE WORK

* Examining how coding style affects the NNs’ ability to understand source code: code written by the same person vs code written by different people
  + Thinking about how the naming of functions and methods affects program understanding
* Building specific NNs that process and understand source code without needing to convert to ASTs or graphs
* Cross Language Understanding: Examining what happens if training happens in one language and testing happens in another language e.g., training in Python and testing with C
* Expand on the outcomes of this project and build a NN that can detect errors (syntax and runtime) in programs
* Expand on the outcomes of this project and build a NN that takes a program and generates a description of what the program is doing.
* Expand on the outcomes of this project and build a NN that finds out the time complexity of a program

##### POTENTIAL APPLICATIONS

* Finding errors (syntax and runtime) in programs, specifically larger programs where it is difficult or impractical for programmers to manually search for errors.
* Finding out what a program is doing e.g., finding out what type of sorting algorithm a program is implementing.

1. BIBLIOGRAPHY