



# **Cognitive Code Analyzer**

**D.J.Thirunayan**  
(Reg. No.: MS20908188)  
**M.Sc. in IT**  
**Specialized in EAD**

Supervisor : Mr. Dilshan De Silva

October 2021

**Department of Information Technology**  
**Faculty of Graduate Studies And Research**  
**Sri Lanka Institute of Information Technology**

# Table of Contents

Table of Contents.....	2
List of Figures .....	3
List of Tables .....	4
Chapter 1 Abstract .....	5
Chapter 2 Introduction .....	6
Chapter 3 Background .....	8
3.1 AST And SBT .....	8
3.2 Computational Complexity Metrics .....	9
3.3 Recurrent Models .....	9
3.4 Source Code Embeddings .....	11
3.5 Encoder-Decoder Architecture .....	14
3.6 Attention .....	15
Chapter 4 Literature Survey .....	17
4.1 Predicting computational runtime complexity .....	18
4.2 Review Generation.....	20
4.2.1 Initial works on source code review generation.....	20
4.2.2 A multi-headed hydranet approach.....	24
4.2.3 An end-to-end deep learning based approach .....	25
Chapter 5 Methodology.....	26
5.1 Data Collection And Preprocessing.....	26
5.1.1 Time complexity prediction dataset .....	26
5.1.2 Review Generation Dataset .....	27
5.1.3 Primary usage of Java source code datasets .....	29
5.1.4 Common complexities in source code datasets .....	30
5.1.5 Data preprocessing methods used for time complexity prediction and dataset structure .....	33
5.2 Architecture .....	35
5.2.1 A hydranet architecture.....	35
5.2.2 The back bone architecture .....	37
5.2.3 Time complexity prediction head .....	39
5.2.4 The review generation head .....	41
5.3 Training .....	43
5.3.1 Training approach .....	43
5.3.2 Loss functions and optimization algorithm.....	45
5.3.3 Training configurations and regularization.....	47

Chapter 6 Experiment .....	48
6.1 Experimentation Process .....	48
6.1.1 Phase 1 – Data collection and analysis .....	49
6.1.2 Phase 2 – Data preprocessing and restructuring .....	49
6.1.3 Phase 3 – Model architecture design and development .....	49
6.1.4 Phase 4 – Model training and hyperparameter tuning .....	50
6.1.5 Phase 5 – Model evaluation .....	50
6.2 Ablation tests .....	51
6.2.1 The SBT traversed AST representation .....	51
6.2.2 Effect on GRU based encoder on performance .....	52
6.2.3 Effect of using code2seq embeddings .....	53
6.2.4 Effect of using attention in the decoder .....	54
Chapter 7 Results .....	55
7.1.2 Metrics .....	55
7.1.3 Effectiveness of Cognitive Code Analyzer in time complexity prediction .....	57
7.1.4 Effectiveness of cognitive code analyzer in review comment generation .....	59
7.1.5 Results of ablation experiments .....	60
Chapter 8 Conclusion And Future Research .....	63
8.1 Conclusion .....	63
8.2 Future research directions .....	63
Chapter 9 References .....	64

## List of Figures

Figure 1 - RNN Architecture .....	10
Figure 2 - LSTM Architecture .....	10
Figure 3 - GRU .....	11
Figure 4 - Architecture of an attention based encoder decoder .....	16
Figure 5 - Overall architecture of CORE .....	22
Figure 6 - Review Dataset Generation Process .....	29
Figure 7 - Generic dataset structure .....	34
Figure 8 - Backbone Architecture .....	37
Figure 9 - Time complexity head architecture .....	40
Figure 10 - Review Generation Architecture .....	42
Figure 11 - Training stage 1 .....	44
Figure 12 – Training stage 2 .....	45
Figure 13 - Model experimentation process .....	48
Figure 14 - Loss curve .....	58
Figure 15 - Weight distribution histogram .....	58

## List of Tables

Table 1 - Identifier types and generic rename templates.....	32
Table 2 - Task specific datasets.....	49
Table 3 - Time complexity model metrics.....	57
Table 4 - Review generation BLEU score performance.....	59
Table 5 - Impact of model performance after removing AST representation from dataset .....	60
Table 6 - Model performace with GRU alternatives .....	61
Table 7 - Precision-Recall comparison after ablating code2seq layer .....	61

## Chapter 1 Abstract

Source code is the building block of any form of software and maintaining efficiency and readability of source code is crucial for the long-term maintainability and usability of any software product. And it is the responsibility of software engineering teams to maintain consistent standards for their source code. The most common approach used by software teams to maintain source code readability and identify bugs is through source code review. Source code review is a process in which when an engineer finishes a project component, functionality, or module, before the developed functionality is released the source code changes in the newly developed functionality are reviewed by another software engineer who is typically more experienced. Although code review was proven to be an effective method for maintaining code consistency, one of the biggest problems in source code review is the amount of time spent by engineers to review code. Maintaining consistent efficiency of source code is an even tougher task because there is no single metric to measure the efficiency of source code. And even metrics like time complexity do not have an algorithmically straightforward method of evaluation from source code.

In this work we propose a “Hydranet” inspired deep learning based model architecture which can effectively learn the underlying patterns in the structure of source code through its syntactic and semantic representations and use the learned representations to perform two primary downstream tasks : generating source code review and predicting time complexity.