

**ENHANCED IN-CONTEXT LEARNING FOR CODE ANALYSIS
WITH COMPACT LARGE LANGUAGE MODELS AND MONTE
CARLO TREE SEARCH**

**AJIBOLA IFEDAYO OLABODE
(22PCG02417)
B.Sc. Computer Science, University of Ibadan**

AUGUST, 2024

**ENHANCED IN-CONTEXT LEARNING FOR CODE ANALYSIS
WITH COMPACT LARGE LANGUAGE MODELS AND MONTE
CARLO TREE SEARCH**

BY

**AJIBOLA IFEDAYO OLABODE
(22PCG02417)**

B.Sc. Computer Science, University of Ibadan

**A DISSERTATION SUBMITTED TO THE SCHOOL OF
POSTGRADUATE STUDIES IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE AWARD OF THE MASTER OF
SCIENCE (M.Sc.) DEGREE IN BIOINFORMATICS IN THE
DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES,
COLLEGE OF SCIENCE AND TECHNOLOGY, COVENANT
UNIVERSITY, OTA, OGUN STATE, NIGERIA**

AUGUST, 2024

ACCEPTANCE

This is to attest that this dissertation is accepted in partial fulfilment of the requirements for the award of the Master of Sciences in Bioinformatics degree in the Department of Computer and Information Sciences, College of Science and Technology, Covenant University, Ota, Nigeria.

Miss Adefunke F. Oyinloye

(Secretary, School of Postgraduate Studies)

Signature and Date

Prof. Akan B. Williams

(Dean, School of Postgraduate Studies)

Signature and Date

DECLARATION

I declare that **Ajibola, Ifedayo Olabode**, with matriculation number **22PCG02417**, conducted this research entitled “**Enhanced In-Context Learning for Code Analysis with Compact Large Language Models and Monte Carlo Tree Search**”. It was carried out under the supervision of Dr. Azubuike Ezenwoke. Concepts of this research project are the results of the research carried out by Ajibola, Ifedayo Olabode, and other researchers' ideas have been fully recognised.

AJIBOLA, IFEDAYO OLABODE

Signature and Date

CERTIFICATION

This is to certify that this dissertation titled “**Enhanced In-Context Learning for Code Analysis with Compact Large Language Models and Monte Carlo Tree Search**”. is original research carried out by **AJIBOLA, IFEDAYO OLABODE (22PCG02417)** in the Department of Computer and Information Sciences, College of Science and Technology, Covenant University, Ota, Ogun State, Nigeria under the supervision of Dr. Azubuike Ezenwoke. We have examined and found this work acceptable as part of the requirements for the award of Master of Science (M.Sc.) in Computer Science.

Dr. Azubuike Ezenwoke

(Supervisor)

Signature and Date

Prof. Olufunke Oladipupo

(Head of Department)

Signature and Date

Prof. Zacchaeus Omogbadegun

(External Examiner)

Signature and Date

Prof. Akan B. Williams

(Dean, School of Postgraduate Studies)

Signature and Date

DEDICATION

I dedicate this work to God Almighty for His grace, wisdom and knowledge given to me throughout my master's degree Programme.

ACKNOWLEDGEMENTS

I express my deepest gratitude to God Almighty for His unwavering guidance, wisdom, and strength throughout this season of my life. This journey would not have been possible without His divine intervention. I extend my sincere appreciation to my supervisor, Dr. Azubuike Ezenwoke, for his invaluable guidance, sacrifices, and unwavering support. His dedication has been instrumental to the success of this work. I am also deeply grateful to my friends and colleagues, Ademolu Ajao, Julius Odunuga, Otavie Okuoyo, and Rifin Ashuza, for their insightful discussions, care, and support throughout my academic journey. To my family, Mr. Tajudeen Ajibola, Mrs. Atoke Ajibola, and Oyindamola Ajibola, thank you for your constant support and encouragement. Your belief in me has been a driving force behind my academic pursuits. I would also like to acknowledge Mr. Bayo Atekoja for his support throughout this endeavor. Finally, I thank all those who have supported me directly or indirectly, your contributions are deeply appreciated.

TABLE OF CONTENTS

CONTENTS	PAGES
COVER PAGE	i
TITLE PAGE	ii
ACCEPTANCE	iii
DECLARATION	iv
CERTIFICATION	v
DEDICATION	vi
ACKNOWLEDGEMENTS	vii
TABLE OF CONTENTS	viii
LIST OF FIGURES	xiii
LIST OF TABLES	xiv
ABBREVIATIONS	xv
ABSTRACT	xvi
CHAPTER ONE	1
1.1 Background to the Study	1
1.2 Statement of the Problem	4
1.3 Aim and Objectives of the Study	4
1.4 Research Methodology	5
1.4.1 Objective 1: Design and Implement an MCTS-Based Algorithm Tailored for Code Understanding and Reasoning in Compact LLMs.	5
1.4.2 Objective 2: Evaluate the Effectiveness of the Algorithm in Code Documentation Generation Tasks Using Open-Source Datasets	5
1.4.3 Objective 3: Benchmark the Performance of the Algorithm Against Existing State-of-the-Art Methods on Established Code-Related Tasks.	6
1.5 Significance of the Study	7
1.6 Scope of the Study	7
1.7 Organization of the Dissertation	8

CHAPTER TWO	9
2.1 Large Language Models in Code Analysis	9
2.2 Retrieval Augmented Generation (RAG) and the Challenges of Long Contextual Dependencies	9
2.3 Code Generation with Large Language Models	11
2.3.1 GPT-4o (Generative Pre-trained Transformer 4-Omni)	11
2.3.2 Codex	11
2.3.3 AlphaCode	12
2.3.4 LLaMA (Large Language Model for Applications)	12
2.3.5 PLBART (Programming Language BART)	13
2.3.6 CodeT5	13
2.4 Long Context Windows in Large Language Models: Challenges and Considerations	14
2.4.1 The "Lost in the Middle" Problem	14
2.4.2 The "Needle in the Haystack" Challenge	15
2.4.3 Computational Complexity and Resource Requirements	15
2.4.4 Consistency and Coherence	15
2.5 Monte Carlo Tree Search (<i>MCTS</i>) in AI	16
2.5.1 Selection	16
2.5.2 Expansion	17
2.5.3 Simulation	17
2.5.4 Backpropagation	17
2.6 Optimization Techniques in Large Language Models	17
2.6.1 Pruning	17
2.6.2 Quantization	18
2.6.3 Knowledge Distillation	18
2.6.4 Transfer Learning	18
2.7 Challenges in Code Generation with LLMs	19
2.7.1 Code Correctness	19
2.7.2 Generalization Across Programming Languages	19
2.7.3 Handling Complex Contexts	19
2.7.4 Efficiency and Resource Constraints	19
2.7.5 Bias in Code Generation	20
2.7.6 Ethical and Legal Implications	20

2.8	Related Works	20
2.9	Gaps and Opportunities	23
2.10	Summary	23
CHAPTER THREE		24
3.1	Research Design	24
3.1.1	Repository Knowledge Graph Construction	24
3.1.2	Integration of MCTS	25
3.1.3	Experimental Setup	25
3.1.4	Evaluation and Comparative Analysis	25
3.2	Development of the LLM-Powered Agent Architecture	25
3.3	Experimental Setup	26
3.3.1	Hardware Configuration	26
3.3.2	Datasets	27
3.3.3	Baseline Models	27
3.4	Evaluation Metrics	29
3.5	Implementation of MCTS	31
3.5.1	Steps of MCTS Implementation	31
3.5.2	Algorithmic Complexity	34
3.6	Comparative Analysis	34
3.6.1	Evaluation Metrics	34
3.6.2	Discussion of Trade-Offs	35
CHAPTER FOUR		36
4.1	Performance of the MCTS-Enhanced LLaMA 3.1 8b Model	36
4.1.1	Accuracy	37
4.1.2	Precision	37
4.1.3	Recall	37
4.1.4	F1-Score	37
4.1.5	Memory Usage	38
4.1.6	Resource Utilization	38
4.2	Comparative Analysis with State-of-the-Art API-Based Models	38
4.2.1	Accuracy Comparison	39
4.2.2	Precision Comparison	40

4.2.3	Recall Comparison	40
4.2.4	F1-Score Comparison	40
4.3	Comparative Analysis with Open-sourced Code Fine-Tuned Models	40
4.3.1	Accuracy Comparison	41
4.3.2	Precision Comparison	41
4.3.3	Recall Comparison	42
4.3.4	F1-Score Comparison	43
4.3.5	Memory Usage (GB)	43
4.3.6	Processing Time (s)	43
4.3.7	Energy Consumption (W)	43
4.3.8	CPU Usage (%)	44
4.4	Challenges and Limitations	44
4.5	Implications for Practice	45
CHAPTER FIVE		46
5.1	Summary of Findings	46
5.1.1	Integration of MCTS	46
5.1.2	Performance Evaluation	46
5.1.3	Comparative Analysis	46
5.2	Challenges and Limitations	47
5.3	Contributions to Knowledge	47
5.3.1	Novel Integration of MCTS with LLaMA 3.1 8b for Code Repository Analysis	47
5.3.2	Comprehensive Performance Evaluation	47
5.3.3	Foundational Work for Future Research	48
5.4	Recommendations for Future Research	48
5.4.1	Exploration of Additional Optimization Techniques	48
5.4.2	Generalization Across Diverse Codebases	48
5.4.3	Automated Parameter Tuning for MCTS	48
5.4.4	Expansion to Additional Code Analysis Tasks	49
5.4.5	Ethical Considerations and Bias Mitigation	49
5.4.6	Longitudinal Studies to Assess Model Performance Over Time	49
5.4.7	Practical Strategies for Integration with Development Tools	49

5.5	Conclusion	49
-----	------------	----

	REFERENCES	51
--	-------------------	-----------

LIST OF FIGURES

FIGURE	TITLE OF FIGURE	PAGES
Figure 2.1:	Retrieval Augmented Generation	10
Figure 2.2:	Lost-in-the-Middle Challenge in Long Context Comprehension	14
Figure 2.3:	Monte Carlo Tree Search	16
Figure 3.1:	Architecture of Agent-Based System	24
Figure 4.1:	Comparative Analysis with State-of-the-Art Models	40
Figure 4.2:	Comparing Precision Performance of LLM Models	43
Figure 4.3:	Compare Recall Performance of LLM Models	43
Figure 4.4:	Comparing F1-Score Performance of LLM Models	44

LIST OF TABLES

TABLE	TITLE OF TABLE	PAGES
Table 3.1:	Baseline Models	27
Table 4.1:	Performance of the MCTS-Enhanced LLama 3.1 8b Model across the various datasets	36
Table 4.2:	Comparative Analysis with State-of-the-Art API-Based Models	39
Table 4.3:	Comparative Analysis with Open-sourced Code Fine-Tuned Models	42

ABBREVIATIONS

Artificial Intelligence	AI
Automatic Software Engineering	ASE
Abstract Syntax Tree	AST
Bidirectional Auto-Regressive Transformers	BART
Bilingual Evaluation Understudy	BLEU
Continuous deployment	CD
Continuous integration	CI
Central Processing Unit	CPU
Gigabyte	GB
Graphical Processing Unit	GPU
Integrated Development Environment	IDE
Longest Common Subsequence	LCS
Large Language Model	LLM
Monte Carlo Tree Search	MCTS
Natural Language Processing	NLP
Retrieval Augmented Generation	RAG
Random Access Memory	RAM
Recall-Oriented Understudy for Gisting Evaluation	ROUGE
Solid-State Drive	SSD
Terabyte	TB
Upper Confidence Bound	UCB
Upper Confidence Bound applied to Trees	UCT

ABSTRACT

Large Language Models (LLMs) demonstrate impressive reasoning abilities, yet their performance can falter when dealing with extensive context lengths. Techniques like Retrieval Augmented Generation (RAG) and Chain-of-Thought prompting seek to bridge this gap, but they face limitations when applied to large code-based contexts due to the complexity of representing inter-object relationships. Monte Carlo Tree Search (MCTS), a heuristic search algorithm, offers a potential solution by aiding LLMs in identifying crucial code repository aspects, thus facilitating downstream tasks. This research focuses on applying MCTS to enhance the performance of "Compact LLMs" - models small enough to run inference on consumer-grade GPUs. Our findings confirm that MCTS indeed boosts performance compared to the baseline Compact LLM. However, these compact models, even with MCTS, still lag behind larger models in performance.

Keywords: large language model, compact LLM, chain-of-thought, monte carlo tree search, in-context learning, code analysis