A Sentiment Analysis of Hate Speech in Philippine Election-Related Posts Using BERT Combined with Convolutional Neural Networks and Model Variations Incorporating Hashtags and ALL-CAPS

A Thesis

Presented to the

Department of Computer Science

College of Information and Computing Sciences

University of Santo Tomas

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science in Computer Science

By

Mendoza, Micah Collette O.

Nadurata, Wayne Gabriel S.

Ortiz, Mark Gabriel E.

Padlan, Joshua Mari L.

Technical Adviser:

Asst. Prof. Charmaine S. Ponay

Dec 2023

Approval Sheet

Thesis Title: A Sentiment Analysis of Hate Speech in Philippine Election-Related Posts Using BERT Combined with Convolutional Neural Networks and Model Variations Incorporating Hashtags and ALL-CAPS

Researchers:

1. **Micah Collette O. Mendoza**
2. **Wayne Gabriel S. Nadurata**
3. **Mark Gabriel E. Ortiz**
4. **Joshua Mari L. Padlan**

In partial fulfillment of the requirements for the degree of **Bachelor of Science in Computer Science,** the thesis mentioned above, has been adequately prepared and submitted by above mentioned researchers. This thesis was duly defended in an oral examination before a duly constituted tribunal with a grade of

Asst. Prof. Janette Sideño
Panel Member

Assoc. Prof. Perla P. Cosme
Panel Member

Mr. Von Guron
Panel Member

Asst. Prof. Cherry Rose Estabillo
Thesis Coordinator

Asst. Prof. Charmaine S. Ponay
Thesis Adviser

Accepted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

Asst. Prof. Jose Seño
Chairperson
Department of Computer Science
College of Information and Computing Sciences

University of Santo Tomas

College of Information and Computing Sciences

Department of Computer Science

Certificate of Authenticity and Originality

We, the authors of this proposed thesis, "**A Sentiment Analysis of Hate Speech in Philippine Election-Related Posts Using BERT Combined with Convolutional Neural Networks and Variant Models Incorporating Hashtags and ALL-CAPS**", hereby certify and vouch that the contents of this research work is solely our own original work; that no part of this work has been copied nor taken without due permission or proper acknowledgment and citation of the respective authors; that we are upholding academic professionalism by integrating intellectual property rights laws in research and projects as requirements of our program.

If found and proven that there is an attempt or committed an infringement of copyright ownership, we are liable for any legal course of action sanctioned by the University and the Philippine laws.

Micah Collette O. Mendoza

Mark Gabriel E. Ortiz

Wayne Gabriel S. Nadurata

Joshua Mari L. Padlan

Program: BS Computer Science

Date: December 1, 2023

UNIVERSITY OF SANTO TOMAS
College of Information and Computing Sciences
Department of Computer Science
2nd Term A/Y 2022-2023

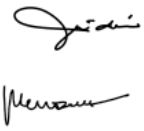
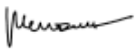
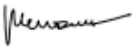
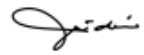
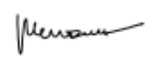
SUMMARY OF COMMENTS, REVISIONS, and RECOMMENDATIONS

Proposed Thesis Title: A Sentiment Analysis of Hate Speech in Philippine Election-Related Tweets Using BERT Combined with Convolutional Neural Networks

Thesis Proposal Defense Date: May 14, 2023

Proponents: Mendoza, Micah Collette O.
Nadurata, Wayne Gabriel S.
Ortiz, Mark Gabriel E.
Padlan, Joshua Mari L.

Expected Date of Presentation of Revisions: May 20-24, 2023

Area / Topic / Section / Chapter	Status During the Presentation	Suggestions/ Recommendations	Suggested / Recommended by	Status [Complied / Not Complied/Partially Complied	Signature of Thesis Adviser / Panel Member / Coordinator, if Complied
Chapter 1: • Scope and Limitations p. 16 Chapter 3: • Hypotheses and Assumption p. 55-56	Chapters 1-3 Initial Draft	Four comparisons: 1. BERT-CNN vs fastText CNN 2. BERT-CNN + Hashtags vs fastText CNN 3. BERT-CNN + ALL-CAPS vs fastText CNN 4. BERT-CNN +	Asst. Prof. Janette Sideño Assoc. Prof. Perla P. Cosme	Complied	
• Research Method p. 57 • Research Design p. 61		Hashtags + ALL-CAPS vs fastText CNN			
Chapters 1 and 3	Chapters 1-3 Initial Draft	Italicize fastText when referring to the model of the base study	Assoc. Prof. Perla P. Cosme	Complied	
Chapter 1: Background of the Study p. 5	Chapters 1-3 Initial Draft	Provide an explanation for employing BERT	Assoc. Prof. Perla P. Cosme	Complied	
Chapter 1: Significance of the Study p. 17	Chapters 1-3 Initial Draft	Emphasize the importance of BERT-CNN	Asst. Prof. Janette Sideño	Complied	
Chapter 1: Conceptual Framework p. 11	Chapters 1-3 Initial Draft	Update conceptual framework by including ALL-CAPS processing	Asst. Prof. Janette Sideño, Assoc. Prof. Perla P. Cosme	Complied	
Chapter 1: Objectives of the Study p. 15	Chapters 1-3 Initial Draft	Append "of the Study" to "Objectives"	Assoc. Prof. Perla P. Cosme	Complied	

Asst. Prof. Janette E. Sideño
Panel Member 1

Date: 5-29-2023

Asst. Prof. Charmaine S. Ponay
Thesis Adviser

Date: 5/31/23

Assoc. Prof. Perla P. Cosme
Panel Member 2

Date: May 23, 2023

Asst. Prof. Cherry Rose R. Estabillo
Thesis Coordinator

Date: June 1, 2023

UNIVERSITY OF SANTO TOMAS
College of Information and Computing Sciences
Department of Computer Science
1st Term A/Y 2023-2024

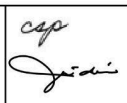
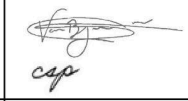
SUMMARY OF COMMENTS, REVISIONS, and RECOMMENDATIONS

Proposed Thesis Title: A Sentiment Analysis of Hate Speech in Philippine Election-Related Posts Using BERT Combined with Convolutional Neural Networks and Model Variations Incorporating Hashtags and ALL-CAPS

Thesis Defense Date: December 11, 2023

Proponents: Mendoza, Micah Collette
Nadurata, Wayne Gabriel
Ortiz, Wayne Gabriel
Padlan, Joshua

Expected Date of Presentation of Revisions: December 13-15, 2023

Area / Topic / Section / Chapter	Status During the Presentation	Suggestions/ Recommendations	Suggested / Recommended by	Status [Complied / Not Complied/Partially Complied	Signature of Thesis Adviser / Panel Member / Coordinator, if Complied
Thesis Title	Chapters 1-5 Defense Final Manuscript	Check if there may be a need to modify the title to highlight the different models	Assoc. Prof. Perla P. Cosme	Complied	
Chapter 1 • Statement of the Problem p. 15	Chapters 1-5 Defense Final Manuscript	Revise SOP 3	Asst. Prof. Janette Sideño	Complied	
Chapter 1 • Scope and Limitations p. 16	Chapters 1-5 Defense Final Manuscript	Omit on the limitation the mention of X's API	Mr. Von Guron	Complied	
Chapter 4 • System Architecture p. 75	Chapters 1-5 Defense Final Manuscript	Improve on the System Architecture (and its documentation) to highlight the innovative parts of the proposed model	Assoc. Prof. Perla P. Cosme	Complied	
Chapter 4 • Summary of Findings p. 180	Chapters 1-5 Defense Final Manuscript	Provide a Summary of Findings under Chapter IV	Assoc. Prof. Perla P. Cosme	Complied	
Chapter 5 • Summary of Findings p. 184	Chapters 1-5 Defense Final Manuscript	Restate Summary of Findings - include findings with results	Asst. Prof. Janette Sideño	Complied	
Chapter 5 • Summary of Findings p. 184 • Conclusions p. 191	Chapters 1-5 Defense Final Manuscript	Classify all responses/results of CH5 from SOP	Asst. Prof. Janette Sideño	Complied	

Asst. Prof. Janette Sideño Panel Member 1 Date: 12-29-2023	Asst. Prof. Charmaine S. Ponay Thesis Adviser Date: 12/15/23
Assoc. Prof. Perla P. Cosme Panel Member 2 Date: December 30, 2023	Asst. Prof. Cherry Rose Estabillo Thesis Coordinator Date: 12.31.2023
Mr. Von Guron Panel Member 3 Date: December 15, 2023	

Acknowledgment

This thesis owes its completion to the invaluable support of numerous individuals, and the researchers extend their profound gratitude to those who played a pivotal role in this journey.

The researchers wish to express heartfelt appreciation to the following individuals for their unwavering support, assistance, and significant contributions to the successful completion of this study.

Foremost, gratitude is extended to family and friends whose belief in the researchers and encouragement were constant sources of motivation throughout the entire process.

To Asst. Prof. Charmaine S. Ponay, their thesis adviser, whose time, patience, and invaluable advice greatly influenced the academic research. The imparted knowledge will forever remain an integral part of the study, and without her guidance, this paper would not have come to fruition.

To Asst. Prof. Cherry Rose Estabillo, their thesis coordinator, for providing the opportunity to undertake the study and for offering valuable recommendations and suggestions to overcome the challenges encountered.

To Asst. Prof. Janette E. Sideño, Assoc. Prof. Perla P. Cosme, and Mr. Von Guron, the thesis defense panelists, for their understanding and constructive feedback, which contributed to strengthening the integrity of the study.

Lastly, heartfelt appreciation is directed to the Almighty Father, whose blessings bestowed the researchers with life, wisdom, and the passion to see the research through to its conclusion.

Abstract

As the number of people who use X continually increases, the same thing is true for hate speech. A pressing need exists for automatic detection of posts that promote hate speech. The datasets gathered and validated from the base study by Argañosa et al. (2022) were used to categorize posts as either hate or non-hate and classify as either positive, negative or neutral with the utilization of Conventional Neural Networks.

The partitioning of the labeled data into training and testing sets adhered to a ratio scheme: 70%-30%, 80%-20%, and 90%-10%. The model of this study, BERT-CNN, had an overall better performance in comparison to the base study, fastText CNN. Particularly, among the three splits, the BERT-CNN model for binary classification without the features of Hashtags and ALL-CAPS with the 90:10 split achieved the best performance with an accuracy of 93.55%, precision of 93.59%, and F1-score of 93.55%.

For multi label classification, the BERT-CNN model demonstrated its optimal performance when incorporating hashtags, specifically with the 90:10 split, achieving an accuracy of 69.14%, precision of 68.44%, recall of 68.40%, and an F1-score of 67.41%. The innovative use of BERT word embeddings paired with CNN proved to excel in classifying Philippine election-related posts as hate or non-hate.

Table of Contents

The Problem and Its Background..... 1
A. Introduction..... 1
B. Background of the Study..... 2
C. Theoretical Framework..... 6
D. Conceptual Framework..... 10
E. Statement of the Problem..... 14
F. Objectives of the Study..... 15
G. Scope and Limitations..... 16
H. Significance of the Study..... 17
Review of Related Literature and Studies..... 22
A. Natural Language Processing (NLP)..... 22
B. Sentiment Analysis..... 23
C. Convolutional Neural Networks (CNN)..... 30
D. Bidirectional Encoder Representations from Transformers (BERT)..... 33
E. BERT-CNN..... 37
F. Hate Speech..... 40
G. Synthesis of Related Literatures and Studies..... 43
Research Design and Methodology..... 59
A. Hypotheses and Assumptions..... 59
B. Research Method..... 60
C. Research Design..... 61
a. Information Gathering..... 61
b. Research Analysis..... 62
c. Designing the Procedures and Algorithms..... 62
d. Implementation of the Model..... 63
e. Testing and Debugging..... 63
f. Evaluation..... 64
D. Specifications of Methodology..... 64
a. Research Instrument..... 64
b. Sources of Information..... 65
E. Sampling and Data Gathering..... 66
F. Statistical Treatment..... 66
a. Confusion Matrix..... 67
b. Accuracy..... 69
c. Precision..... 69
d. Recall..... 70

e. F1 Score 71
f. Macro Average (Multi Label Classification) 71
g. McNemar's Test with Holm-Bonferroni Correction 72
Presentation and Analysis of Data 74
A. System Architecture 74
B. Description of Modules and Interfaces 77
B.1. Preprocessing 78
B.2. Splitting 80
B.3. BERT Word Embeddings 81
B.4. Binary and Multi Label CNN 82
B.5. Evaluation 83
C. Test Cases 84
C.1. BERT-CNN without Hashtags and ALL-CAPS 85
C.2. BERT-CNN with Hashtags and without ALL-CAPS 85
C.3. BERT-CNN without Hashtags and with ALL-CAPS 87
C.4. BERT-CNN with Hashtags and All-Caps 88
D. System Demonstration of Test Data 89
D.1. Main System Process 89
D.1.1. Data Loading and Extraction 89
D.1.2 Preprocessing 92
D.1.2.1 De Identification and URL Removal 92
D.1.2.2 Special Character Processing 93
D.1.2.3. Hashtags Processing 94
D.1.2.4. Normalization (with ALL-CAPS Processing) 95
D.1.2.5. Tokenization 99
D.1.3 Splitting 100
D.1.4. BERT Word Embeddings 101
D.1.5 Binary and Multi Label CNN 101
D.1.6. Evaluation 104
E. Results and Analysis 106
E.1. fastText CNN Binary Classification 106
E.1.1. 70:30 Split 107
E.1.2. 80:20 Split 108
E.1.3. 90:10 Split 110
E.2. fastText CNN Multi Label Classification 111
E.2.1. 70:30 Split 111
E.2.2. 80:20 Split 113
E.2.3. 90:10 Split 114
E.3. BERT-CNN Binary Classification 115
E.3.1. BERT-CNN 116

E.3.1.1. 70:30 Split....117
E.3.1.2. 80:20 Split....119
E.3.1.3. 90:10 Split....121
E.3.2. BERT-CNN with Hashtags....122
E.3.2.1. 70:30 Split....123
E.3.2.2. 80:20 Split....125
E.3.2.3. 90:10 Split....127
E.3.3. BERT-CNN with ALL-CAPS....128
E.3.3.1. 70:30 Split....129
E.3.3.2. 80:20 Split....131
E.3.3.3. 90:10 Split....133
E.3.4. BERT-CNN with Hashtags and ALL-CAPS....134
E.3.4.1. 70:30 Split....135
E.3.4.2. 80:20 Split....137
E.3.4.3. 90:10 Split....139
E.4. BERT-CNN Multi Label Classification....140
E.4.1. BERT-CNN....141
E.4.1.1. 70:30 Split....141
E.4.1.2. 80:20 Split....143
E.4.1.2. 90:10 Split....145
E.4.2. BERT-CNN with Hashtags....146
E.4.2.1. 70:30 Split....147
E.2.2.2. 80:20 Split....149
E.2.2.3. 90:10 Split....151
E.4.3. BERT-CNN with ALL-CAPS....152
E.4.3.1. 70-30 Split....153
E.4.3.2. 80-20 Split....155
E.4.3.3. 90-10 Split....157
E.4.4. BERT-CNN with Hashtags and ALL-CAPS....158
E.4.4.1. 70-30 Split....159
E.4.4.2. 80-20 Split....161
E.4.4.3. 90-10 Split....163
E.5. Text Analysis with BERT Embeddings Integration....165
E.5.1 Candidate Names....165
E.5.2. Predictions....167
E.5.3. Misclassification....168
E.6. Model Comparisons....170
E.6.1. BERT-CNN and fastText CNN Comparison....170
E.6.2. BERT-CNN w/ HT and BERT-CNN w/o HT Comparison....171
E.6.3. BERT-CNN w/ AC and BERT-CNN w/o AC Comparison....172
E.6.4. BERT-CNN w/ HT+AC and BERT-CNN w/o HT+AC Comparison....173

E.6.5. BERT-CNN and fastText CNN Comparison....174
E.6.6. BERT-CNN w/ HT and BERT-CNN w/o HT Comparison....175
E.6.7. BERT-CNN w/ AC and BERT-CNN w/o AC Comparison....176
E.6.8. BERT-CNN w/ HT+AC and BERT-CNN w/o HT+AC Comparison....177
E.7. Statistical Analysis....178
F. Summary of Findings....180
Summary of Findings, Conclusions and Recommendations....183
A. Summary....183
B. Conclusions....191
C. Recommendations....193
References....196

List of Figures

Figure 1.1. Architecture of fastText CNN Model - Hate Speech in Filipino Election-Related Posts: A Sentiment Analysis Using Convolutional Neural Networks (Argañosa et al., 2022)..... 7
Figure 1.2. Architecture of BERT-CNN Model - BERT-CNN: Improving BERT for Requirements Classification using CNN (K. Kaur & P. Kaur, 2023)..... 9
Figure 1.3. Conceptual Framework..... 11
Figure 4.1. System Architecture..... 75
Figure 4.2. Preprocessing Module..... 78
Figure 4.3. Splitting Module..... 80
Figure 4.4. Pre Trained BERT Word Embeddings Module..... 81
Figure 4.5. Binary and Multi label CNN Module..... 82
Figure 4.6. Evaluation Module..... 83
Figure 4.7. Data Loading and Extraction Code Snippet for Binary Classification..... 90
Figure 4.8. Data Loading and Extraction Code Snippet for Multi Label Classification... 90
Figure 4.9. Mention and URL Removal Function..... 92
Figure 4.10. Special Character Processing..... 93
Figure 4.11. Hashtags Processing..... 95
Figure 4.12. Normalization (Lowercase)..... 95
Figure 4.13. ALL-CAPS Processing (Lowercase except ALL-CAPS)..... 96
Figure 4.14. Tokenization..... 99
Figure 4.15. Code Snippet of 70:30 Data Splitting..... 100
Figure 4.16. Code Snippet of 80:20 Data Splitting..... 101
Figure 4.17. Code Snippet of 90:10 Data Splitting..... 101
Figure 4.18. BERT Word Embeddings..... 101
Figure 4.19. CNN Model for Binary..... 102
Figure 4.20. CNN Model for Multi Label..... 103
Figure 4.21. Model Evaluation Metrics..... 105
Figure 4.22. Classification Report Sample..... 105
Figure 4.23. Classification Report (fastText CNN 70:30)..... 107
Figure 4.24. Confusion Matrix (fastText CNN 70:30)..... 107
Figure 4.25. Classification Report (fastText CNN 80:20)..... 108
Figure 4.26. Confusion Matrix (fastText CNN 80:20)..... 109
Figure 4.27. Classification Report (fastText CNN 90:10)..... 110
Figure 4.28. Confusion Matrix (fastText CNN 90:10)..... 110
Figure 4.29. Classification Report (fastText CNN 70:30)..... 111
Figure 4.30. Confusion Matrix (fastText CNN 70:30)..... 112
Figure 4.31. Classification Report (fastText CNN 80:20)..... 113
Figure 4.32. Confusion Matrix (fastText CNN 80:20)..... 113
Figure 4.33. Classification Report (fastText CNN 90:10)..... 114

Figure 4.34. Confusion Matrix (fastText CNN 90:10).....115
Figure 4.35. Train-Test Loss (BERT-CNN 70:30).....117
Figure 4.36. Classification Report (BERT-CNN 70:30).....117
Figure 4.37. Confusion Matrix (BERT-CNN 70:30).....118
Figure 4.38. Train-Test Loss (BERT-CNN 80:20).....119
Figure 4.39. Classification Report (BERT-CNN 80:20).....119
Figure 4.40. Confusion Matrix (BERT-CNN 80:20).....120
Figure 4.41. Train-Test Loss (BERT-CNN 90:10).....121
Figure 4.42. Classification Report (BERT-CNN 90:10).....121
Figure 4.43. Confusion Matrix (BERT-CNN 90:10).....122
Figure 4.44. Train-Test Loss (BERT-CNN w/ HT 70:30).....123
Figure 4.45. Classification Report (BERT-CNN w/ HT 70:30).....123
Figure 4.46. Classification Report (BERT-CNN w/ HT 70:30).....124
Figure 4.47. Train-Test Loss (BERT-CNN w/ HT 80:20).....125
Figure 4.48. Classification Report (BERT-CNN w/ HT 80:20).....125
Figure 4.49. Confusion Matrix (BERT-CNN w/ HT 80:20).....126
Figure 4.50. Train-Test Loss (BERT-CNN w/ HT 90:10).....127
Figure 4.51. Classification Report (BERT-CNN w/ HT 90:10).....127
Figure 4.52. Confusion Matrix (BERT-CNN w/ HT 90:10).....128
Figure 4.53. Train-Test Loss (BERT-CNN w/ AC 70:30).....129
Figure 4.54. Classification Report (BERT-CNN w/ AC 70:30).....129
Figure 4.55. Confusion Matrix (BERT-CNN w/ AC 70:30).....130
Figure 4.56. Train-Test Loss (BERT-CNN w/ AC 80:20).....131
Figure 4.57. Classification Report (BERT-CNN w/ AC 80:20).....131
Figure 4.58. Confusion Matrix (BERT-CNN w/ AC 80:20).....132
Figure 4.59. Train-Test Loss (BERT-CNN w/ AC 90:10).....133
Figure 4.60. Classification Report (BERT-CNN w/ AC 90:10).....133
Figure 4.61. Confusion Matrix (BERT-CNN w/ AC 90:10).....134
Figure 4.62. Train-Test Loss (BERT-CNN w/ HT+AC 70:30).....135
Figure 4.63. Classification Report (BERT-CNN w/ HT+AC 70:30).....135
Figure 4.64. Confusion Matrix (BERT-CNN w/ HT+AC 70:30).....136
Figure 4.65. Performance Measures (BERT-CNN w/ HT+AC 80:20).....137
Figure 4.66. Classification Report (BERT-CNN w/ HT+AC 80:20).....137
Figure 4.67. Confusion Matrix (BERT-CNN w/ HT+AC 80:20).....138
Figure 4.68. Performance Measures (BERT-CNN w/ HT+AC 90:10).....139
Figure 4.69. Classification Report (BERT-CNN w/ HT+AC 90:10).....139
Figure 4.70. Confusion Matrix (BERT-CNN w/ HT+AC 90:10).....140
Figure 4.71. Performance Measures (BERT-CNN 70:30).....141
Figure 4.72. Classification Report (BERT-CNN 70:30).....142
Figure 4.73. Confusion Matrix (BERT-CNN 70:30).....142
Figure 4.74. Performance Measures (BERT-CNN 80:20).....143

Figure 4.75. Classification Report (BERT-CNN 80:20)..... 143
Figure 4.76. Confusion Matrix (BERT-CNN 80:20)..... 144
Figure 4.77. Performance Measures (BERT-CNN 90:10)..... 145
Figure 4.78. Classification Report (BERT-CNN 90:10)..... 145
Figure 4.79. Confusion Matrix (BERT-CNN 90:10)..... 146
Figure 4.80. Performance Measures (BERT-CNN w/ HT 70:30)..... 147
Figure 4.81. Classification Report (BERT-CNN w/ HT 70:30)..... 147
Figure 4.82. Confusion Matrix (BERT-CNN w/ HT 70:30)..... 148
Figure 4.83. Performance Measures (BERT-CNN w/ HT 80:20)..... 149
Figure 4.84. Classification Report (BERT-CNN w/ HT 80:20)..... 149
Figure 4.85. Confusion Matrix (BERT-CNN w/ HT 80:20)..... 150
Figure 4.86. Performance Measures (BERT-CNN w/ HT 90:10)..... 151
Figure 4.87. Classification Report (BERT-CNN w/ HT 90:10)..... 151
Figure 4.88. Confusion Matrix (BERT-CNN w/ HT 90:10)..... 152
Figure 4.89. Performance Measures (BERT-CNN w/ AC 70:30)..... 153
Figure 4.90. Classification Report (BERT-CNN w/ AC 70:30)..... 153
Figure 4.91. Confusion Matrix (BERT-CNN w/ AC 70:30)..... 154
Figure 4.92. Performance Measures (BERT-CNN w/ AC 80:20)..... 155
Figure 4.93. Classification Report (BERT-CNN w/ AC 80:20)..... 155
Figure 4.94. Confusion Matrix (BERT-CNN w/ AC 80:20)..... 156
Figure 4.95. Performance Measures (BERT-CNN w/ AC 90:10)..... 157
Figure 4.96. Classification Report (BERT-CNN w/ AC 90:10)..... 157
Figure 4.97. Classification Report (BERT-CNN w/ AC 90:10)..... 158
Figure 4.98. Performance Measures (BERT-CNN w/ HT+AC 70:30)..... 159
Figure 4.99. Classification Report (BERT-CNN w/ HT+AC 70:30)..... 159
Figure 4.100. Confusion Matrix (BERT-CNN w/ HT+AC 70:30)..... 160
Figure 4.101. Performance Measures (BERT-CNN w/ HT+AC 80:20)..... 161
Figure 4.102. Classification Report (BERT-CNN w/ HT+AC 80:20)..... 161
Figure 4.103. Confusion Matrix (BERT-CNN w/ HT+AC 80:20)..... 162
Figure 4.104. Performance Measures (BERT-CNN w/ HT+AC 90:10)..... 163
Figure 4.106. Classification Report (BERT-CNN w/ HT+AC 90:10)..... 163
Figure 4.107. Confusion Matrix (BERT-CNN w/ HT+AC 90:10)..... 164

List of Tables

Table 2.1. Synthesis Table..... 43
Table 3.1. Confusion Matrix for Binary Classification.....67
Table 3.2. Confusion Matrix for Multi Label Classification.....68
Table 3.3. McNemar's Test Table.....73
Table 4.1. Test Case for BERT-CNN without Hashtags and ALL-CAPS.....85
Table 4.2. Test Case for BERT-CNN with Hashtags and without ALL-CAPS..... 86
Table 4.3. Test Case for BERT-CNN without Hashtags and with ALL-CAPS..... 87
Table 4.4. Test Case for BERT-CNN with Hashtags and ALL-CAPS.....88
Table 4.5. Binary Classification Dataset.....90
Table 4.6. Multi Label Classification Dataset..... 91
Table 4.7. Preprocessed Dataset.....94
Table 4.8. Preprocessed Binary and Multi Label Dataset with Test Cases.....96
Table 4.9. Tokenized Dataset.....100
Table 4.10. Sample Candidate Names with Prediction Values..... 166
Table 4.11. Misclassifications.....168
Table 4.12 Comparison of Performance Measures for fastText.....170
CNN and BERT-CNN models for Binary Classification.....170
Table 4.13 Comparison of Performance Measures for BERT-CNN
with Hashtags and BERT-CNN without Hashtags models for
Binary Classification.....171
Table 4.14 Comparison of Performance Measures for BERT-CNN
with ALL-CAPS and BERT-CNN without ALL-CAPS models for
Binary Classification.....172
Table 4.15 Comparison of Performance Measures for BERT-CNN
with Hashtags and ALL-CAPS and BERT-CNN without Hashtags
and ALL-CAPS models for Binary Classification..... 173
Table 4.16 Comparison of Performance Measures for fastText CNN..... 174
and BERT-CNN models for Multi Label Classification..... 174
Table 4.17 Comparison of Performance Measures for BERT-CNN
with Hashtags and BERT-CNN without Hashtags models for
Multi Label Classification..... 175
Table 4.18 Comparison of Performance Measures for BERT-CNN
with ALL-CAPS and BERT-CNN without ALL-CAPS models for
Multi Label Classification..... 176
Table 4.19 Comparison of Performance Measures for BERT-CNN
with Hashtags and ALL-CAPS and BERT-CNN without Hashtags
and ALL-CAPS models for Multi Label Classification..... 177

Table 4.20. McNemar's Test with Holm-Bonferroni for BERT-CNN and.....178
fastText CNN models for Binary Label Classification.....178
Table 4.21. McNemar's Test with Holm-Bonferroni for BERT-CNN and.....179
fastText CNN models for Multi Label Classification..... 179

Chapter I

The Problem and Its Background

This chapter presented the introduction, background of the study, theoretical frameworks, conceptual framework, statement of the problem, objective of the study, scope and limitations, and the definition of terms used in the study.

A. Introduction

The term political hate speech is commonly referred to as an act of marginalizing or dehumanizing groups or individuals for their political ideology, beliefs or affiliation. This involves the use of coded phrasing with underlying discriminatory intent, persuasion to boycott, ‘cancel’ and even dissemination of false or misleading information. This act can be done by anyone at any time with more frequent action during elections or other political campaigns. These certain hate speeches are often engaged on various platforms such as social media which are being exploited to spread discrimination and misinformation (OHCHR, n.d).

Relevantly, X, formerly known as Twitter, served as a great platform to express filtered and unfiltered thoughts and ideas of both facts and opinion. In line with this, X became the netizen’s choice for voicing political opinions, criticisms, and perceptions. This means that posts, formerly known as tweets, composed of hate speeches and offensive posts towards an opposing political faction or party are often visible and may lead to negative consequences which is a considerable problem as this may turn into threats or harassment that may generate fear within the receiving end. In fact, posts

composed of hate speeches were almost at an all time high during the 2022 Philippine Election period due to its polar nature which caused a massive divide. During this time, supporters of a particular candidate would berate each other, call each other names, belittle the opposing candidate, or even much worse.

With the voluminous data available, it is a challenge to detect the amount of hate speech automatically. Computational models have been created to automatically detect hate speech such as the study from the University of Santo Tomas (Argañosa et. al., 2022) with the detection of Filipino election-related posts with *fastText* CNN. However, despite the high accuracy of roughly 83%, there is still room for improvement. As such, the researchers aimed to answer the following questions with respect to the problem of the study: Can the combination of BERT with the CNN model improve the performance of the existing model proposed by the authors of the base study in detecting hate speech in Philippine election-related posts? To what extent can the model reach its optimal performance? How can this be improved? What insights can be derived from the analyzed text data with the utilization of BERT embeddings?

B. Background of the Study

As the number of people who use X continually increases, the same thing is true for hate speech. A pressing need exists for automatic detection of posts that promote hate speech. This is possible through sentiment analysis, which is also known as opinion mining. It is a technique in NLP which extracts and classifies subjective text such as emotional statements, opinions or personal rants. It analyzes these texts into categories — positive, negative or neutral, as a feature. This technique is useful for identifying opinion

polarities in many applications such as customer feedback, product reviews or even opinionated discussions. It also involves analyzing and monitoring social media activities and gathering user posts to understand emotion and opinionated sentiment (Nanli, 2012).

In detecting hate speech, the posts can be classified as hate or non-hate for binary classification, and positive, neutral, or negative for multi label classification. Similar to this, a study was conducted to analyze the components of hate speech in posts with topics on the 2016 Philippine Presidential Elections (Solitana & Cheng, 2021). The authors of the study made use of K-means clustering paired with Principal Component Analysis (PCA) to segregate and cluster hate speech targets away from the general groupings. With 55% of hate targets and post groupings, results show that clustering the targeted hate speech from the groupings based on annotation alone is not a suitable approach for nearest neighbors in K-means while the inclusion of different lexical terms found in posts proved as a potential classification improvement. Another classification study was conducted on classifying subjective Filipino texts with the use of feature-based approaches namely, Naïve Bayes, Bagging, Multilayer Perceptron, and Random Forest Tree, to classify subjective lexicons using strong and weak opinionated terms and evaluating each of their performance with a 10-fold cross validation (Regalado & Cheng, 2012). The results of this study show that both Bagging and Random Forest Tree models outperformed the other models which resulted in 63.2941% classification accuracy. A study achieved higher results with the use of both Standard Recurrent Neural Networks (Standard RNN) and Bi-directional Recurrent Neural Networks (Bi-directional RNN) on analyzing sentiments in collected from posts in the time of November 2013 to January 2014 relating to the Typhoon Yolanda event in the Philippines (Imperial et al., 2019). The

authors of the study also considered including the annotations or hashtags in posts during their preprocessing stage. As a result of this, together with sentiment analysis as well as modified hyperparameters and configurations, a rate of 87.12% classification accuracy of Standard RNN and a rate of 87.69% classification accuracy of Bi-directional RNN were achieved.

Students from the University of Santo Tomas (Argañosa et al., 2022) proposed a study that analyzes hate speech in Filipino election-related posts through sentiment analysis. They made use of Convolutional Neural Networks (CNN) to perform the two aforementioned types of classifications. The results of the authors' study show that *fastText* CNN model was able to surpass the TF-IDF Feed Forward Neural Networks (FFNN) model, which is the model used in the base study, in terms of accuracy, precision, recall, and F-measure. Even though the model performed with a relatively high accuracy of approximately 83%, improvements can still be made with the architecture's hyperparameters and components of the posts used for preprocessing to contribute with the classification accuracy. To improve the model's performance, fine-tuning the hyperparameters and layers used in the CNN are necessary.

Additionally, increasing the size and diversity of the training dataset can help improve the model's learning capacity and enable it to better discern token relationships and classification patterns. The wider context provided by more posts accumulated can help the model fit the labeling of various posts in general. However, recent changes in X API policies have significantly impacted the researchers' ability to access additional data beyond a particular date. X stated on February 9, 2023, that free access to the X API (v2 and v1.1) will be discontinued and replaced with a paid basic tier. These changes have

introduced challenges in gathering additional data for enhancing the model's learning capacity through a larger and more diverse training dataset.

Retaining hashtags especially on political posts were also found to be useful in determining the sentiments expressed as they can already provide insight into the political viewpoint as stated in a study authored by Alfina et al. (2017). This makes it important to correctly classify the sentiment polarities of political posts, as the presence of hashtags can make the process more accurate when annotating the dataset. Additionally, using ALL-CAPS as another feature can provide some indication of the sentiment expressed in the text. Capitalization, specifically using ALL-CAPS, is often associated with emphasis, strong emotions, or expressing intensity (Chan et al., 2018). In political discussions, the use of ALL-CAPS can signal passion, anger, or a heightened sentiment towards a particular political figure, issue, or ideology. By implementing these improvements, the model's performance in sentiment analysis of Filipino election-related posts can be further enhanced. A similar study on detecting hate speech in Philippine election-related posts (Baluyut, 2022) included hyperparameter tuning in their model flow as they believed it was the method to arrive at the best performance with their models which included tree-based, linear, and multilayer perceptron builds.

In a study conducted by Malik et al. (2021), the combination of BERT word embeddings with CNN produced exceptional results. The authors explored different embedding techniques, such as fastText and BERT word embeddings, and inputted the results into deep neural network algorithms including Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), and CNN. Among these algorithms, the combination of BERT embedding and CNN demonstrated the highest performance, outperforming the

other approaches. In line with this, the study aimed to improve the accuracy of the system primarily through the use of BERT word embeddings that will be combined with CNN model for sentiment analysis. BERT will replace *fastText* word embeddings of Argañosa et al. (2022) model.

The development of hate speech detection through sentiment analysis is a continuous effort that requires the incorporation of various techniques and methodologies. While the studies discussed above have shown promising results, further research and improvement were necessary to effectively combat hate speech in social media platforms. This study improved upon the base study's model, *fastText* CNN, by implementing BERT-CNN models. Four (4) BERT-CNN models were used in this study: (1) BERT-CNN, (2) BERT-CNN with Hashtags, (3) BERT-CNN with ALL-CAPS, and (4) BERT-CNN with Hashtags and ALL-CAPS.

C. Theoretical Framework

This part of the study included extensive theoretical information, models, and cases from preceding researchers and experts. The models and theories were carefully picked to align in this field and research topic.

As mentioned in the background of the study, a group of students from the University of Santo Tomas (Argañosa et al., 2022) proposed a thesis study that analyzes hate speech in Filipino election-related posts through the use of *fastText* CNN. The authors made note of conducting sentiment analysis on bilingual posts in relation to the country’s 2022 political elections and classifying them as positive, neutral, and negative as well as categorizing them into hate and non-hate. The preprocessed data was then split

and tested with the models. Figure 1.1 showed the framework for the architecture, from processing the posts to training the models with the chosen algorithms, and testing the trained models.

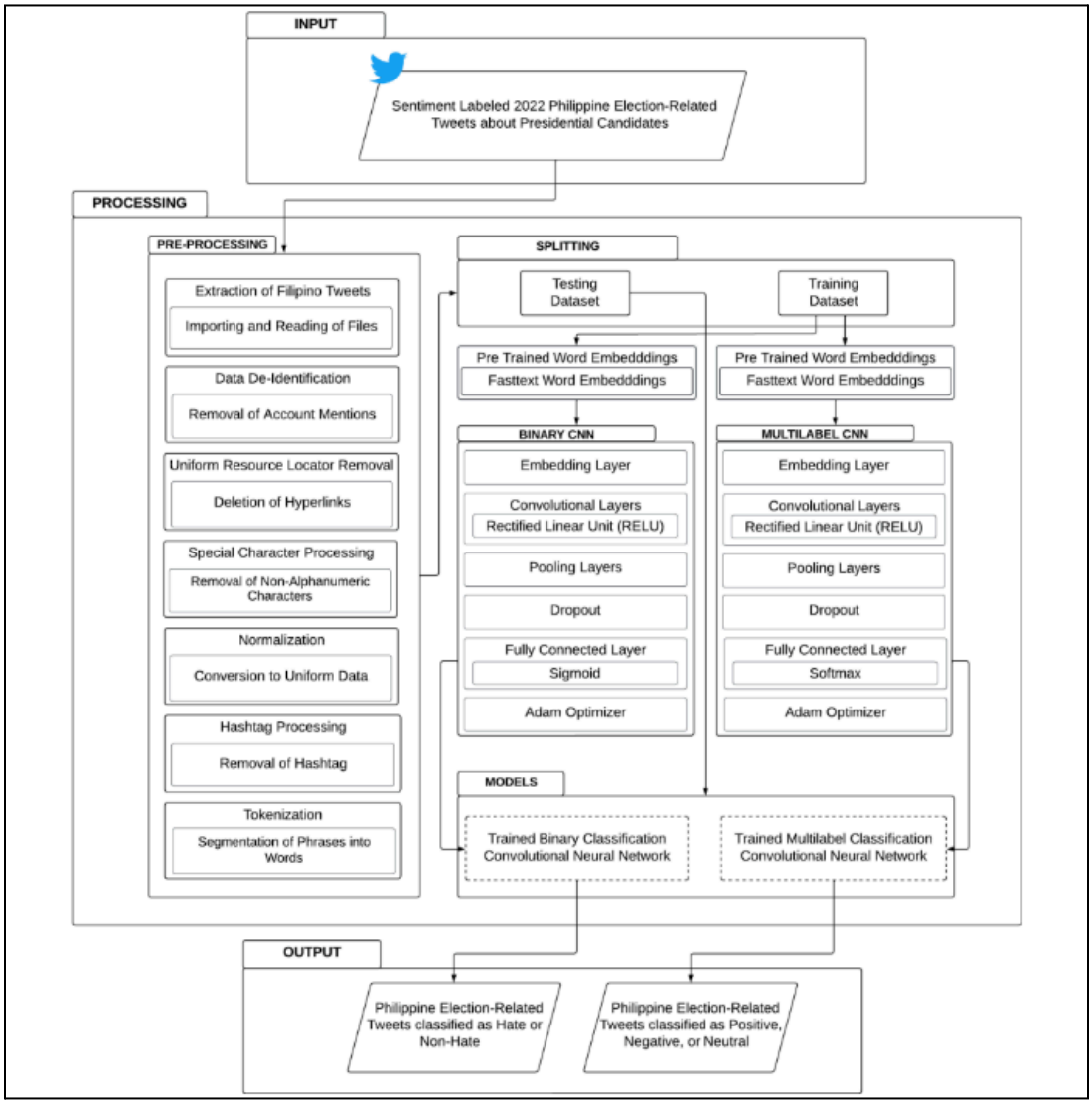


Figure 1.1. Architecture of fastText CNN Model - Hate Speech in Filipino Election-Related Posts: A Sentiment Analysis Using Convolutional Neural Networks (Argañosa et al., 2022)

Bidirectional Encoder Representations from Transformers (BERT) is a pre-trained language representation model from unlabeled text which can be fine-tuned with only one additional layer and perform a variety of NLP tasks such as sentiment analysis and translating language. One of the prime features with this model is its ability to understand sentences and the context behind it instead of relying on the preceding or succeeding words. This makes BERT one of the most powerful algorithms to date. (Devlin et al., 2018). Combining BERT word embedding with CNN yielded outstanding results in a study by Malik et al. (2021). The authors employed embedding techniques such as fastText and BERT word embeddings and fed in the results into deep neural network algorithms namely, MLP, LSTM, and CNN. The combination of BERT embedding and CNN had the best performance among the three algorithms.

K. Kaur and P. Kaur (2023) also recognized BERT's powerful language modeling capabilities as well as CNN's extraction of rich features from input text. BERT overcomes the drawback of both Word2Vec and GloVe, which is the inability to find contextual information from a sequence of related words. In other words, it is capable of taking into account the context while retaining the word sequence. Boosting the accuracy of a classifier, such as CNN, is also possible through the use of BERT. As a result, the authors proposed the BERT-CNN model. The model comprised four key layers: embedding layer, convolutional layer, pooling layer, and output layer. In generating word vectors of input sequence, BERT was employed which also established an initial input matrix. Using the input vector matrix as a basis, the convolutional layer generates feature maps. These feature maps were then passed to the pooling layer, which extracts the maximum value

feature vectors. The feature vectors are then fed into a fully connected layer to facilitate classification. Figure 1.2 illustrated the proposed model's architecture.

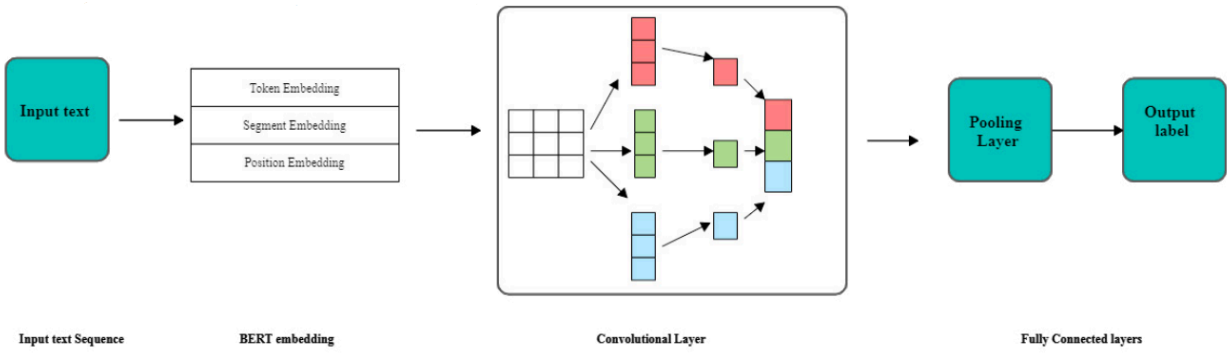


Figure 1.2. Architecture of BERT-CNN Model - BERT-CNN: Improving BERT for Requirements Classification using CNN (K. Kaur & P. Kaur, 2023)

As hashtags may have representational value when it comes to classifying posts, hashtags embedded in posts can be kept. In the study of Kaviani and Rahmani (2020), the authors spotlight the pivotal role of hashtags especially on social media platforms like X. As such, they introduced a novel hashtag recommendation model called EmHash, which utilizes a neural network based on BERT embeddings. They have observed in the previous hashtag recommendation models, the underutilization of BERT embeddings, which takes context into account while producing word vectors. The integration of BERT embedding in EmHash not only advances hashtag recommendation but also has implications for sentiment analysis. The model's ability to consider context in word representations enhances its capacity to accurately capture and respond to the sentiment

expressed in posts, making it a valuable tool for applications that involve understanding and analyzing user emotions in online communication.

D. Conceptual Framework

This section of the thesis presented a conceptual framework that has been developed based on relevant theories and studies, with necessary modifications to fit the chosen topic. The framework provides a clear flow of the model and serves as the theoretical basis of the study. It helps the researchers to organize their ideas and concepts, and to identify the key components and relationships among them. The conceptual framework is essential for providing a structured approach to the study, and for ensuring that the research objectives are met.

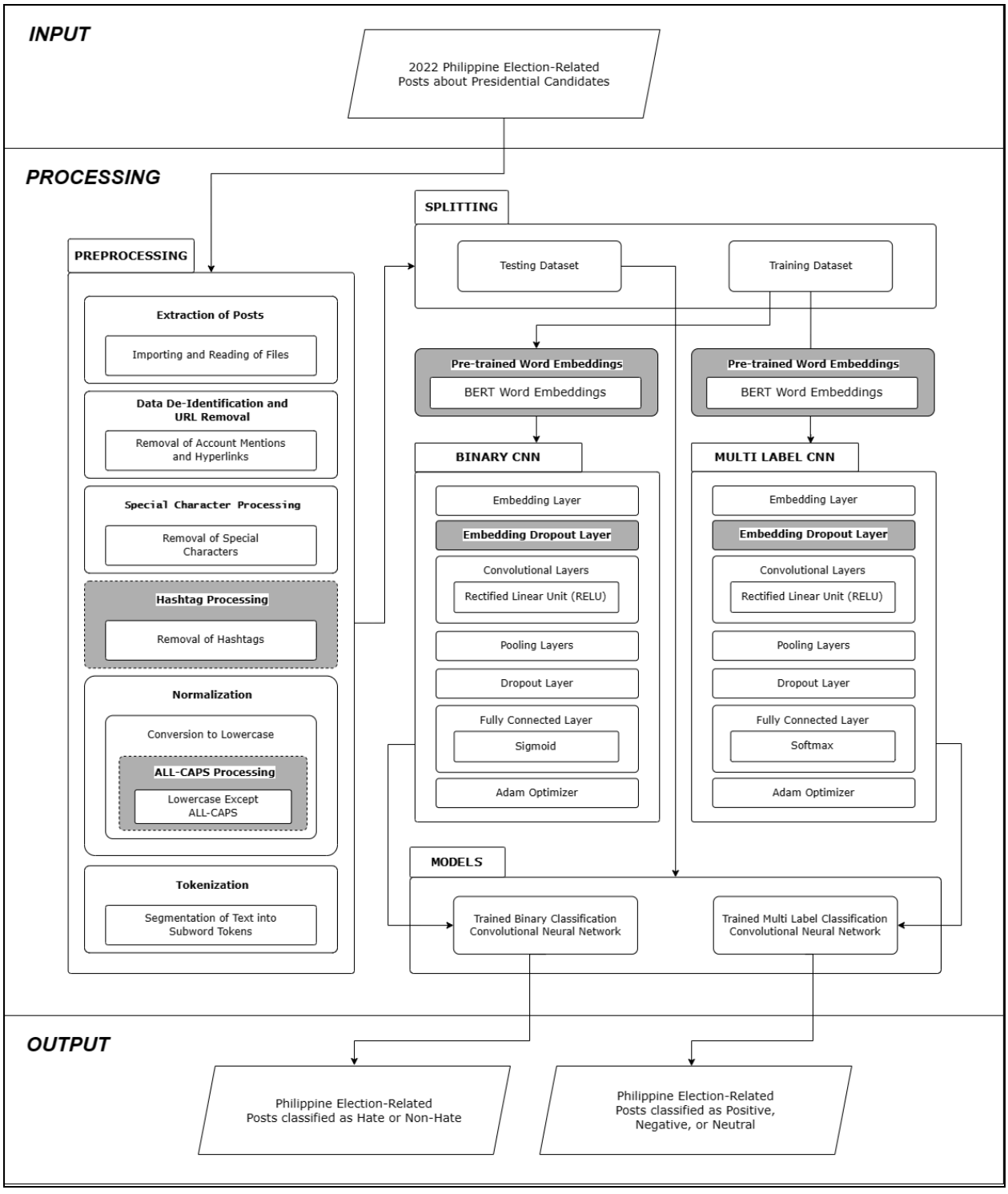


Figure 1.3. Conceptual Framework

The posts related to the 2022 Philippine elections and pertaining to presidential candidates were collected from the study of Argañosa et al. (2022). Two types of datasets will be used in this study: binary classification dataset and multi label classification dataset.

The data was imported into a programming environment like Jupyter Notebook to execute Python codes, where it was prepared for further analysis. The data then underwent several a series of preprocessing steps including the removal of X account mentions, URLs, special characters (also numerics), and candidate names. Unlike the approach used in the study of Argañosa et al. (2022) where English and Filipino stopwords were removed, this study opted to retain them. In contextual language models like BERT, retaining stopwords helps preserve the context of a statement, allowing the model to better understand the intent. The approach further diverged as this study incorporated hashtags and ALL-CAPS in sentiment analysis. Additionally, to address variations of the BERT-CNN model due to the inclusion of hashtags, ALL-CAPS, or both, there were variations in the subsequent preprocessing steps as well. Specifically, for hashtag processing, hashtags were removed for the base BERT-CNN and BERT-CNN with ALL-CAPS. However, they were retained for other models utilizing hashtags. For ALL-CAPS processing, it was under the normalization step. Every text was converted to lowercase format but the casing for ALL-CAPS were preserved for models that utilizes them. Using hashtags and ALL-CAPS in sentiment analysis provided a more comprehensive understanding of the sentiment expressed in the text, particularly in a political context where emotions and specific topics are often emphasized. Finally, the textual data was tokenized into individual words to facilitate its use in vectorization.

After the preprocessing of data, the binary and multi label classification datasets were split into their corresponding training and testing datasets. The splits were 70:30, 80:20, and 90:10.

The training datasets were used as an input for the pre-trained BERT models to generate word embeddings. Several studies have shown the effectiveness of incorporating BERT word embeddings into CNN models for various NLP tasks. For instance, in a study on sentiment analysis of commodity reviews, Dong et al. (2020) showed that a CNN model with BERT embeddings achieved the best performance. Similarly, in a study on offensive speech identification, Safaya et al. (2020) demonstrated that a CNN model with BERT embeddings outperformed BERT and CNN models on their own.

The word embeddings are fed into the embedding layer of the CNN models, followed by an embedding dropout layer to reduce the risk of overfitting. These embeddings were then passed through a series of convolutional layers, which used a set of filters to extract features from the sequence of embeddings. After the convolutional layers, a Rectified Linear Unit (ReLU) activation function was applied. This will be followed by pooling layers, which reduce the dimensionality of the feature maps. The output of the pooling layers went through the dropout layer which reduced the risk of overfitting. The output was fed into fully connected layers, which used the learned features to make predictions on the input text. For binary classification, the sigmoid function was utilized, whereas the softmax function was used for the multi label classification. The CNN models made use of the Adam optimizer as the optimization algorithm.

The testing datasets were used for the trained BERT-CNN models to perform sentiment analysis on Philippine election-related posts. Classifications of hate and non-hate for binary classification, and positive, negative, and neutral for multi label classification were determined. The classification results were then used for the evaluation of the model's performance in computing for accuracy, precision, recall, and F1 score.

E. Statement of the Problem

As mentioned in the study of Argañosa, et al. (2022) from the background of the study, the authors' findings show that their model architecture in detecting hate speech in Filipino election-related posts can still be improved. Specifically, while the model presented in the study achieved a relatively high accuracy of approximately 83%, there is still room for improvement in their architecture and preprocessing. The inclusion of hashtags and ALL-CAPS provided a more comprehensive understanding of the sentiments. Therefore, the problem this study aimed to address was how to improve the performance of sentiment analysis in Filipino election-related posts, particularly in detecting hate speech, by employing the BERT-CNN model and improvement of preprocessing pipeline.

With this, the researchers aimed to answer the following questions:

1. How will the BERT-CNN model perform in terms of accuracy, precision, recall, and F1 score in the sentiment analysis of Philippine election-related posts?
2. How will the BERT-CNN model compare to the *fastText* CNN model in terms of the performance measures on the sentiment analysis of Philippine election-related posts?
3. How does the utilization of BERT embeddings enhance the BERT-CNN model's performance in sentiment analysis of Philippine election-related posts?

F. Objectives of the Study

The main objective of this study was to determine whether BERT along with CNN can be effective in identifying political hate speech on X. At the same time, the study aimed to compare the performance of its model with the one proposed by Argañosa et al. (2022).

The specific objectives of this study are the following:

1. To determine the accuracy, precision, recall, and F1 score of the BERT-CNN model in the sentiment analysis of Philippines election-related posts.
2. To compare the results of the BERT-CNN model to *fastText* CNN in terms of the performance of sentiment analysis of Philippine election-related posts

3. To analyze and interpret the insights on data derived from the utilization of BERT embeddings in the BERT-CNN model for sentiment analysis of Philippine election-related posts.

G. Scope and Limitations

The study focused on determining the overall performance of detecting hate speech in Philippine election-related posts by utilizing the BERT-CNN model instead of the base study's CNN with *fastText* word embeddings. The researchers of the study tested whether there were improvements in the system's performance by implementing the BERT-CNN model as replacement for the base study's model. The BERT-CNN model alone and the BERT-CNN model in combination with features were compared with the base study's *fastText* CNN model. The said combinations were the following: (1) BERT-CNN with Hashtags, (2) BERT-CNN with ALL-CAPS, and (3) BERT-CNN with Hashtags and ALL-CAPS.

The training and testing data were posts consisting of textual data with or without Hashtags and/or ALL-CAPS. The posts that were collected and used were limited to the labeled dataset of posts collected from Github Repository that contributed to the study by Argañosa et. al. (2022). Consequently, the study relied on the available data from the Github Repository as the primary source for labeled posts.

The composition of posts were restricted to English, Filipino, and a combination of all mentioned languages. The dates of the posts spanned around during the Philippine pre-elections from the 8th of October 2021 to the 7th of May 2022. The study did not

cover topics of the Philippine elections other than contents with interest of the presidential candidates. The gathered data were only considered as a representation of the portion of hate speech in the country, and the study did not present the data as the main root of hate speech for citizens and communities in the country. The prime programming language used to create, train, and test the model was through the third version of Python with the latest being 3.11.1. The related study and literature chosen provided information and figures that were not published longer than fifteen (15) years.

The chosen machine learning methods used were limited to combination of the BERT model with CNN, specifically for the improvement of the model used in the base study conducted by Argañosa et al. (2022). The researchers of the study focused on the analysis of the hate speech to be found in 2022 Philippine election-related posts.

H. Significance of the Study

The study aimed to fill the existing gaps in the field of Computer Science, specifically in detecting hate speech from users' posts using sentiment analysis. This research was intended to advance the state of previous studies, particularly in the area of Natural Language Processing (NLP). BERT combined with CNN was especially useful in sentimental analysis as it is an open source model and can establish whole sentences or phrases in deeper meaning or context and weigh them dynamically (Lutkevich, 2020). The following benefited from this study:

- **Social Media Platforms** - Facebook, X, and Instagram are some social media platforms that can use the outcomes of the study to improve their content

moderation policies. Hate speech is a significant concern on social media, especially during sensitive periods like elections. Employing the BERT-CNN model improved their ability to detect and mitigate hate speech and other forms of abusive content from their platforms. This helped improve the user experience, create safer and more inclusive online environments, and protect users from online harassment and abuse.

- **Government and Policymakers** - Government agencies and policy makers greatly benefited from the study's emphasis on the BERT-CNN model in detecting hate speech in election-related posts. Policy makers can gain valuable insights into the prevalence and impact of hate speech during elections, enabling them to formulate effective strategies and policies to counteract its negative effects. The BERT-CNN model's ability to analyze sentiments in text with contextual understanding empowered governments to proactively address hate speech, promote healthy political discourse, and create a safer online environment for citizens to engage in constructive discussions and democratic participation.
- **Public and Users** - The general public and social media users can benefit from the study's outcomes by experiencing a safer and more respectful online environment. The BERT-CNN model's ability to detect hate speech helped minimize the spread of harmful content and mitigate the potential negative effects on individuals and communities. Users can engage in more constructive discussions and interactions while feeling protected from hate speech.
- **Researchers and Academics** - The study filled existing gaps in the field of Computer Science, particularly in the area of detecting hate speech from users'

posts using sentiment analysis. Researchers and academics can benefit from the study by gaining insights into the effectiveness of the BERT-CNN model for analyzing sentiment in text and hashtags. This study provided a novel approach for detecting and analyzing hate speech along with the advancement of the previous studies' state in NLP. As a result, more advanced methods and tools for analyzing and interpreting language data can be developed.

- **Future Researchers** - Future researchers can also benefit from the study's findings and recommendations. They can build upon the study's methodology and findings to improve the accuracy and effectiveness of hate speech detection and sentiment analysis in other contexts or languages. For instance, they can explore the use of BERT-CNN, feature extraction techniques, and data preprocessing approaches to enhance the performance of the hate speech detection model.

I. Definition of Terms

The following terminologies used in this research are the following:

- **Accuracy** - The expected distance between the predicted variable and the target variable. It was defined as the measurement of the models' quality of precision of the results of the study.

- **BERT** - stands for Bidirectional Encoder Representations from Transformers. A deep learning based model. It is a non-directional approach to generate a language model and provides a prediction based on the decoded input. This model was used to replace the *fastText* word embeddings in assigning different meanings to word inputs.

- **CNN** - stands for Convolutional Neural Networks. It is an artificial neural network paired with a deep learning algorithm to analyze and classify text as input. This model served as a paired algorithm to the BERT word embeddings in the study.

- **Hashtags** - A feature from X composed of words or phrases before a hashtag or pound symbol ('#') to relate a user's list of posts or threads into one topic to make it easier for all other users to search or discover. The study included hashtags as a feature to the BERT-CNN model which was compared with the *fastText* CNN model in terms of accuracy and overall performance.

- **Hate Speech** - A form of harmful language which aims to degrade and discriminate against a group or an individual for their ideas and principles. It was also presented as the main target to be classified in the whole study.

- **NLP** - stands for Natural Processing Language. It is a method or algorithm which supports the process of simulating structured and unstructured text in human language to a computer to analyze and understand its meaning. It was used as the primary language identification method to the study for detecting hate speech in Philippine election-related posts.

- **Sentiment Analysis** - A method in NLP which processes subjective or opinionated text and categorizes them based on the author's emotion whether it is positive, negative, or neutral. In the development of the study, it served as the basis for classifying posts into *hate* or *non-hate*.

- **Post** - A specific form of post from the social media platform called X which hosts a certain number of characters to express a user's thoughts. This was utilized as the target form of data to be gathered for the study.

- **Word Embeddings** - A common method of Natural Language Processing (NLP) representation (to vectors). This method assigns meanings to an encoded word, and each carries a meaning assigned the same way. This method was utilized for the BERT model as it replaces *fastText* word embeddings to the study.

Chapter II

Review of Related Literature and Studies

In this chapter, a comprehensive overview and analysis of the existing literature and studies related to the topic of the research was presented after the thorough and in-depth search done by the researchers.

A. Natural Language Processing (NLP)

In computer science and artificial intelligence (AI), an area of research that is referred to as natural language processing, focuses on how to process natural languages like English. In most cases, this processing is converting natural language into data (numbers) that a computer can utilize to understand the outside world. Moreover, natural language text that reflects this perspective of the world can occasionally be produced using it. Because a natural language processing system typically requires numerous steps of processing, where natural language enters at one end and processed output exits at the other, it is frequently referred to as a pipeline (Lane et al., 2019).

Eisenstein (2018) stated that designing and analyzing computational representations and algorithms for the processing of natural human language is the main emphasis of NLP. It aims to develop new computational abilities that revolve on human language, such as information extraction from texts, language translation, question-answering, conversational interaction, and so forth. Although fundamental linguistic knowledge may be necessary for completing these various activities, success is ultimately determined by whether or not the work is completed.

The task of automatically evaluating whether a text unit, such as a sentence or document, reflects a subjective or objective opinion or viewpoint is known as subjectivity classification in NLP. While objective text units communicate factual information or describe occurrences without expressing any personal viewpoints, subjective text units express personal sentiments, ideas, or beliefs. A study of feature-based subjectivity classification of Filipino text was conducted by Regalado and Cheng (2012). Classifying Filipino texts with the use of feature-based approaches namely, Naïve Bayes, Bagging, Multilayer Perceptron, and Random Forest Tree. Different features such as part-of-speech (POS), subjectivity, term presence, and frequency information, were included for the classification process. The data was sourced from the editorial sections of three Filipino newspapers. Crowd-sourcing was conducted to tag each sentence in the editorials as either objective or subjective, wherein votes were used for subjectivity tagging. As there are no available lexicons for opinion or sentiment analysis in Filipino, the authors constructed a lexicon that would assist in subjectivity classification specifically for the Filipino language. Subjective lexicons were classified using strong and weak opinionated terms and evaluating each of their performance with a 10-fold cross validation. The results of this study show that both Bagging and Random Forest Tree models outperformed the other models which resulted in 63.2941% classification accuracy

B. Sentiment Analysis

The field of study known as sentiment analysis, sometimes known as opinion mining, examines people's opinions, sentiments, assessments, attitudes, and emotions about entities and their attributes as they are expressed in written text. The entities can be

things like goods, services, businesses, people, occasions, problems, or subjects. The area of the problem represented by the field is enormous. Sentiment analysis today encompasses a wide range of terms with slightly varied definitions, including sentiment analysis, opinion mining, opinion analysis, opinion extraction, sentiment mining, subjectivity analysis, affect analysis, emotion analysis, and review mining (Liu, 2020).

In a study conducted by Alfina, et al. (2017), the goal is to determine whether utilizing hashtags can be beneficial in classifying the polarity of political posts. With this, the authors proposed the use of hashtags as a feature in classification so that the model can automatically annotate a dataset based on the number of positive and negative hashtags which is referred to as the SentiHT feature. In order to do this, they created a list of extracted hashtags which will then be manually annotated if it is positive, negative, neutral, or irrelevant. The original posts that contain hashtags will then be altered by appending “HT_POS” or “HT_NEG” at the end of the text whenever it is positive or negative. Hashtags with neutral polarity will be simply extracted, on the contrary, if the hashtags have irrelevant polarity, there will be no extraction nor tag addition done. Next, a separator “$$” was used in order to separate the text from the hashtag. Finally, hashtags with uppercase letters will then be extracted as two terms. As mentioned, these hashtags will then be used as a feature to classify the polarity of posts. The study employs binary classification with positive and negative classes.

Furthermore, the authors used three features to classify sentiment polarity which are: the number of positive and negative hashtags (SentiHT), number of positive and negative words (SentiLex), and unigram. To identify the most effective machine learning algorithm for determining sentiment polarity in the datasets they used, a comparison of

classification accuracy using four different algorithms: Naïve Bayes (NB), Support Vector Machine (SVM), Logistic Regression (LR) and Random Forest (RF), was conducted.

Results show that the SentiHT feature has the most significant contribution in determining sentiment polarity, provided that the dataset mostly includes sentiment hashtags. In sentiment classification, the SentiHT feature as well as the automatically labeled dataset with SentiHT, boasted an accuracy of more than 95%. The unigram feature is outperformed by SentiHT paired with NB, SVM, or LR algorithms, except for RF. Since SentiHT's accuracy employing the four algorithms is essentially the same, hence it is advised to use NB, which has the lowest computing time (1.97 seconds). Through the experiments and evaluations conducted in the study, the authors were able to determine that hashtags as a feature can be beneficial in sentiment analysis.

Mehta, et al. (2019) authored a study wherein they proposed models for sentiment analysis that classify posts as well as compare them with each other in terms of performance. Before the experiments were conducted, they made use of regular expressions and the NLTK library in Python in order to clean or preprocess the data. Through the use of the previously mentioned techniques and libraries, the data was cleaned by removing stop words, unwanted notations or symbols, and repetitive words; at the same time the mispelt words in the dataset were already corrected. Once this was completed, they trained and used nine machine learning models namely, Naive Bayes, Support Vector Machine, Decision Tree, Max Entropy, XGBoost, Random Forest, Convolutional Neural Network, Long-Short Term Memory, and Multilayer Perceptron. The results show they have a varying degree of performance when it comes to classifying

posts with the Convolutional Neural Network model, together with GloVe embeddings, being the best which had an accuracy of 79%. The accuracy achieved by the Support Vector Machine was also comparable.

Posts in regards to the 2019 Indonesian presidential election were analyzed using sentiment analysis in the study by Hidayatullah, et al. Since this study aims to obtain the best possible algorithm or model for this given context, various deep neural network algorithms were used in training the dataset such as Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), CNN-LSTM, Gated Recurrent Unit (GRU)-LSTM and Bidirectional LSTM. At the same time, traditional machine learning algorithms were also used such as Support Vector Machines (SVM), Logistic Regression (LR), and Multinomial Naïve Bayes (MNB), in comparison with the authors' deep learning model. Term Frequency-Inversed Document Frequency (TF-IDF) was both utilized and not utilized as a feature with these three algorithms when comparing the accuracy result.

The results of this study show that SVM had the highest accuracy of 84.04% compared to LR and MNB. It was also found that the use of TF-IDF contributed to the improvement of these models' accuracy. On the other hand, the Bidirectional-LSTM model performed the best out of all the deep learning algorithms tested as it is 84.60% accurate. The SVM has lower 0.56% accuracy than Bidirectional LSTM. This proved that Bidirectional LSTM outperformed every model in the study. At the same time, it is also worth mentioning that other deep learning models still performed well with the CNN model being 84.05% accurate.

Roy and Ojha (2020) conducted a study which aims to compare the performance of deep learning models namely Google BERT, attention based Bidirectional LSTM, and Convolutional Neural Networks (CNNs) in regards to sentiment analysis of posts. These models were trained with fine-tuned embeddings using the SemEval-2016 X dataset. Both CNN and Bidirectional-Attention based LSTM make use of pretrained GloVe embeddings. Sentiments are categorized into neutral, negative, or positive.

After conducting the experiments, the authors concluded that BERT with an accuracy of 64.1%, outperformed both Bi-Attentive LSTM and CNN models with 60.02% and 59.2%, respectively. Like the preceding study, it has also been emphasized that compared to machine learning techniques, deep learning models are remarkably effective and accurate in the analysis of sentiments.

In the study conducted by Boquiren et al. (2022), various experiments were conducted to evaluate the effectiveness of using Backward Slang as a feature to determine sentiment polarities in posts. Notably, the developed models did not require any complex modules such as sentiment treebanks or feature engineering for the extraction of special features. Instead, pre-trained word vector representations were used for all versions of the models. Long-Short Term Memory (LSTM) and Bidirectional LSTM (Bi-LSTM) were chosen as the deep learning architectures to extract contextual information from both forward and backward dependencies in feature sequences. To identify the optimal set of parameters, both of these models have undergone hyperparameter tuning through Random Search algorithm. Additionally, for each resulting model, two different train sets were employed—one with backward slang and one without. This results in four different models. A test set containing backward slang

will be used to validate the parameters that work best. The models will then be validated using performance metrics.

The Bi-LSTM models outperformed the LSTM models, especially in terms of recall score. The authors of the study emphasized how it is critical to take the classification accuracy and F1 score into account, when evaluating model performance. On both versions of Bi-LSTM, the difference between accuracy and AUC is 3.45%, while there is a 3.56% difference when it comes to the F1 score. Bi-LSTM obtained a lower value in difference, compared to the differences seen on LSTM. LSTM obtained a 4.28% difference in accuracy and AUC, and a 4.39% in F1 score. This could be an indicator that Bi-LSTM helped reduce the misclassification of labeled posts.

The models were able to classify sentiment polarities with outstanding performance metrics in spite of the complexity of Filipino posts, particularly in the world of politics where sarcasm and satire posts are frequently used. The study also came to the conclusion that performance metrics as well as overall sentiment scores improves through the addition of Backward Slang as a feature. Although Backward Slang can indicate both positive and negative opinions about a post, it has been found that the expressiveness of negative opinions in sentiment analysis is increased more than that of positive opinions when these features are present.

Another study was conducted by Imperial et al., (2019) in order to identify sentiments of posts made before, during, and after the onslaught of Typhoon Yolanda (November 1, 2013 to January 31, 2014). In order to achieve this, the authors made use of both Standard Recurrent Neural Networks (Standard RNN) and Bi-directional Recurrent

Neural Networks (Bi-directional RNN). The inclusion of annotations or hashtags in posts during their preprocessing stage was also considered by the authors of the study. The study generated models with various hyperparameters and found that the best-performing models achieved high accuracy in both fine-grained and binary classification tasks. Specifically, the standard RNN achieved 81.79% accuracy in fine-grained classification while the bidirectional RNN achieved 87.69% accuracy in binary classification. The main objective of the study was to raise awareness about social involvement and empathy among Filipinos during times of calamity. The analysis of the posts revealed that the majority of posts (51.1%) expressed positivity and support towards the victims, while 19.8% expressed negativity such as sadness, despair, and hate towards corrupt officials. The remaining 29% of posts were neutral and consisted of updates from local news stations, relief operation announcements, donation drives, and citizen observations.

Kaviani and Rahmani (2020) conducted a proposal that analyzes the process of BERT word embeddings with the recommended hashtag feature using neural networking. This approach is speculated to have a positive impact on text embeddings which holds relevance in semantic similarity. The proponents of the study made a system which gathers user posts from X and preprocesses the data to collect hashtags and transform them into vectors. The proponents' model then predicts the hashtags based on the text's semantic analysis and finds the most similar hashtags to the text based on cosine similarity using Fine-Grain prediction.

The prediction results show that hashtags, especially hashtags clustering, is recommended for predictions and finding semantic similarity. It was also worth noting that the hashtags used in BERT word embeddings and other approaches such as Emhash

improve output performance, and may also hinder prediction with its number of varieties. This makes hashtags in NLP models both a weakness and strength point.

In the study conducted by Chan et. al. (2018), an analysis of special orthographic characteristics which filter through in social mea platforms such as Twitter or X. This includes the characteristic of capitalization positioning in text. The objective of the study was to analyze and illustrate the behavior of capitalization and other orthographic characteristics and its impact on sentiment analysis. To analyze capitalization with sentiment analysis, their team gathered a dataset on X and transformed the data into tokens using the *TweetTokenizer* toolkit. All text or words with capitalization or all capitalized were counted and analyzed to find patterns.

Results show that text with capitalization on Twitter or X shows a pattern which were labeled as *meaningful* capitalization, which is when capitalization is used on text with intent and expressive value, rather than uses for convenience such as abbreviations. In their study, *meaningful* capitalization in particular did hold an impact on sentiment analysis and functions as a conversational cue to clarify underlying meaning over text-based style communication.

C. Convolutional Neural Networks (CNN)

The Convolutional Neural Network (CNN) was utilized in the sentiment analysis on English posts related to the "Turkey Crisis 2018" topic. The resulting CNN classifier model was then compared to the Naïve Bayes Classifier (NBC) model to determine which provided better accuracy in sentiment analysis. This was conducted in the study Sunarya et al. (2019). The sentiment analysis process began with data retrieval, followed

by data classification using TextBlob to categorize posts as having positive, negative, or neutral sentiment.

The CNN classifier model was built using the Keras Functional API model and had 3 convolutional layers, each with 100 kernel filters. The kernel size was adjusted to the n-gram concept used in each layer, including bigram (2), trigram (3), and fourgram (4). The activation function used in the convolutional layer was ReLu. To extract the maximum value from each filter, three 1D max pooling layers were used in the model architecture. A fully connected layer with dropout was employed to process the output from the max pooling layer, with a total of 128 neurons. Finally, the output layer consisted of 3 neurons with the softmax activation function.

After undergoing training and evaluation, the CNN classifier model achieved an accuracy rate of 89%. During the test data testing process, the model achieved an accuracy rate of 88%. These accuracy results were then compared to those of the Naïve Bayes classifier model, which had an accuracy rate of 78%. Based on this accuracy comparison, it can be concluded that the classifier model utilizing a deep learning algorithm performs better in sentiment analysis than the NBC classifier model.

In another study by Alim (2021), sentiment analysis has been used to analyze natural language processing through machine learning methods. However, in Indonesia, there have been few studies conducted using machine learning techniques with specific datasets. Machine learning often results in low accuracy when applied in a broader context, so the deep learning method was developed to enhance the accuracy of sentiment analysis. The study conducted experiments to demonstrate that CNN models

outperformed machine learning methods. The model categorized sentiments as neutral, positive, or negative. A variety of CNN models were presented with different configuration parameters to improve accuracy performance, including the number of convolutional layers, number of filters, and filter size.

To analyze the model's performance, Indonesian-Sentiment-Analysis-Dataset, which consists of 10,806 posts, was used with the Word2Vec model for Indonesian as a word vector representation. The CNN models were trained on 80% of the dataset and tested on the remaining 20% of the dataset. When comparing the performance metrics of various CNN models for sentiment analysis, it was found that there were no significant differences in accuracy, precision, recall, and F1 score values among the models. The CNN 12 model performed the best, achieving an accuracy of 81.4%, a precision and recall of 81.1%, and an F1 score of 81.0%. On the other hand, the CNN 1 model had the worst performance with an accuracy of 70.5%, a precision and recall of 70.6%, and an F1 score of 70.5%.

The proposed CNN models' performance was compared with that of machine learning algorithms such as Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and Stochastic Gradient Descent (SGD). The best-performing model was CNN 12, which achieved an accuracy of 81.4% using 7, 4, and 3 filter sizes and 256 filters. The CNN models demonstrated superior accuracy performance when compared to machine learning algorithms, with the maximum accuracy achieved by SGD being 62.1%, while SVM and KNN achieved 61.4% and 52.3% accuracy, respectively, for general sentiment analysis in Indonesian.

D. Bidirectional Encoder Representations from Transformers (BERT)

The widespread use of social media has led to the availability of a massive amount of data created by users, which can be analyzed to determine their emotions and opinions. A study authored by Chiorrini et al. (2021) aims to investigate the improvement of classification accuracy using Bidirectional Encoder Representations from Transformers (BERT), a popular pre-trained language model based on Transformer, for both sentiment analysis and emotion recognition tasks. The study proposed two BERT-based architectures for text classification, which are fine-tuned on datasets specifically designed for these tasks. The BERT-Base model was used in this study, in both the uncased and cased versions. This model has 12 encoders, each composed of 8 layers: 4 multi-head self-attention layers and 4 feed forward layers. The uncased version converts text to lowercase and ignores accents during the tokenization process. A fully connected layer and a softmax layer were added by the authors to the BERT-Base model for classification, with the only difference between the sentiment and emotion classifiers being the number of neurons in the last softmax layer, which corresponds to the number of classes, i.e., 3 for sentiment analysis and 4 for emotion recognition. The performance of the models will be evaluated on real-world post datasets.

The BERT model in the uncased version achieved an accuracy of 89% and an F1 score of 89% for emotion recognition. On the other hand, the cased version achieved an accuracy of 90% and an F1 score of 91%, indicating slightly better performance. For sentiment analysis, the uncased and cased BERT have comparable performance with an accuracy of 92% and an F1 score of 92%, indicating that the cased version does not offer any improvement over the uncased one.

The BERT models achieved high levels of accuracy, with 92% for sentiment analysis and 90% for emotion recognition. The authors concluded that BERT's language modeling capability plays a crucial role in achieving good text classification.

Previous natural language processing (NLP) approaches utilizing models such as Word2Vec, convolutional neural network (CNN), recurrent neural network (RNN), and bidirectional long short-term memory (BiLSTM) have been limited in their ability to capture the deeper context of words. In contrast, BERT is capable of encoding all input data, which in this case is the entire sentence, simultaneously, enabling it to take into account the inputs before and after a given word, thereby building a better contextual understanding of the word. In contrast, Word2Vec is limited in its ability to understand a word's meaning across two different contexts and may produce varying output vector values depending on its usage. In a study by Bello, et al. (2023), the BERT model was used along with other models such as CNN, RNN, and BiLSTM, in classifying posts using sentiment analysis. The authors used six different datasets, a combination of various types of posts, to train their model.

The results of the study indicated that the combination of BERT with the aforementioned models demonstrated exceptional performance compared to the results of the models alone or with the use of Word2Vec. BERT-CNN, BERT-RNN, and BERT-BiLSTM yielded high accuracy and F-measure of 93% and 95%, respectively. On the contrary, when Word2Vec was used with CNN, RNN, and BiLSTM, lower accuracy rates were obtained, ranging from 48% to 57%. Similarly, lower F-measure scores were also observed, ranging from 56% to 64%. These results further highlight the significance

of BERT's ability to significantly contribute to achieving accurate text classification, especially in the domain of sentiment analysis for social media platforms such as X.

Several studies have shown the effectiveness of BERT in NLP tasks. However, the lack of publicly available Filipino post datasets regarding fire reports on social media has hindered the development of classification models for Filipino posts. To address this, a study was conducted by Mingua et al. (2021) which aimed to design and implement a system that can classify Filipino posts using different pre-trained BERT models. Using a dataset of 2,081 posts that contain fire-related posts, the authors created a model that can exclusively organize Filipino posts and compared the accuracy of different fine-tuned pre-trained BERT models. The results indicate a significant difference in accuracy among the pre-trained BERT models. The BERT Base Uncased WWM model performed the best, achieving a test accuracy of 87.50% and a train loss of 0.06 during training of the dataset, while the BERT Base Cased WWM model was the least accurate, with a test accuracy of 76.34% and a train loss of 0.2. These results suggest that the BERT Base Uncased WWM model is a reliable model for classifying fire-related posts in Filipino. However, the accuracy of the model may vary depending on the size of the dataset.

Profanities can be used in an aggressive or non-abusive manner, depending on the situation. In a study authored by Galinato et al. (2023), 14,000 X post samples were scraped using SNScrape, and each one was manually classified as abusive or not. Next, using a previously developed Tagalog BERT model, they improved its context-based profanity classification and censorship of Tagalog texts. The authors created the Tagalog X Profanity dataset (TTP), which consisted of 13,746 Tagalog posts that were manually annotated for profanity content. The dataset had a high inter-annotator agreement rate of

96%, based on the results of Fleiss' Kappa coefficient, indicating a high level of agreement between the annotators. To evaluate the model, they employed a Confusion matrix, Matthew's Correlation Coefficient (MCC), Precision, Recall, and Multi-class F1 score.

The original BERT base model from the study of **Devlin et al. (2018)**, which is trained only in English language, cannot be used directly for the study. To address this limitation, the authors utilized an open-source BERT Tagalog base-cased model, which was pre-trained specifically for Tagalog language by Cruz and Cheng (2019). This Tagalog model has the same architecture and parameters as its English counterpart. To create the proposed BERT model, the authors used the pre-trained tokenizer and word embeddings from the Tagalog model as feature vectors, with the help of HuggingFace's Transformers library. Finally, the proposed BERT model was developed and trained using the PyTorch library. The model used the baseline architecture of the pre-trained BERT Tagalog model, but with the addition of a dropout layer of 0.5 and a softmax layer to fine-tune it for classifying abusive and non-abusive profanity usage. Prior to training the model, the TTP dataset was split into a train, validation, and test set in an 80:10:10 ratio. The proposed Tagalog BERT base model was then trained using the hyperparameters: Cross Entropy Loss as the loss function, Adam as the optimizer, 0.00001 learning rate, and batch size of 64.

Results showed that the model's overall accuracy was 86% and its precision and recall is 88%. In the non-abusive and abusive groups, it received F1 scores of 88% and 84%, respectively, with sample support of 786 and 589. Also, the model obtained an MCC score of .70, indicating a strong connection between the predicted label and the

actual label in classification. It has been concluded by the authors that the TTP dataset was effectively annotated for context-based profanity classification and that the proposed BERT model was successful in detecting and censoring Tagalog profanity in text media content.

E. BERT-CNN

A study by Dong et al. (2020) highlighted the importance of sentiment analysis in improving the quality of commodities and influencing the purchasing decisions of consumers. However, the accuracy of existing sentiment analysis models needed to be improved. Therefore, the authors proposed a BERT-CNN model that improves the accuracy of sentiment analysis in commodity reviews. The model involved constructing a BERT model and using a representation layer to encode the review texts. Then, a CNN semantic extraction layer was used to extract local features of the review text vectors, while a BERT semantic extraction layer was used to extract global features. The features were then fused using a semantic connection layer, and sentiment analysis was performed using the sentiment classification layer.

The authors used both CNN and BERT models to extract local and global features of review texts respectively. The features extracted by these two complementary models were then combined to address a limitation of using a single CNN model, which was the neglect of semantic relations within the context of review texts. By integrating the features extracted by CNN and BERT, the model was able to effectively uncover semantic relations between short sentences in long texts. The experiment was conducted on a mobile phone review dataset in JD Mall.

The BERT-CNN model was evaluated by comparing it with the original CNN model and BERT model. To ensure the accuracy and objectivity of the experimental results, all three models were executed 10 times on the same training and test sets. Precision, recall rate, and F1 value were calculated as the final model results. The experimental results showed that the BERT-CNN model outperformed the BERT and CNN models by about 14.4% and 17.4% in terms of F1 value. Furthermore, both the BERT-CNN model and BERT model show higher F1 values compared to the CNN model, both in the training set and the test set. The reason for this is that BERT and BERT-CNN are unsupervised pre-training models that can accurately connect semantic contexts and mine accurate sentence meanings in commodity review texts, without the need for artificial intervention. The attention mechanism of these models can also accurately identify the semantic logic between sub phrases in each review text. Moreover, the difference between the F1 value of the BERT-CNN model in the training set and that in the test set is very small, indicating that the model has good generalization ability.

In another study by Safaya et al. (2019), the authors combined pre-trained BERT and CNN models for text analysis. BERT, which is a deep bidirectional transformer model developed by Devlin et al. in 2019, provided the knowledge for the model, while CNN, a widely-used approach for text classification tasks, was used to analyze the text. This combination of models has been proven to be more effective than using either BERT or CNN alone, as shown in this study and a previous one by Li et al. in 2019. The study emphasized the importance of using pre-trained language models for downstream tasks, such as identifying offensive language. The dataset given for this task (Zampieri et al.,

2020) included collections of posts that have been labeled as either Offensive (positive) or Non-offensive (negative).

The experiments conducted in this study used four different pre-trained BERT models, including three language-specific models for Greek, Turkish, and Arabic, as well as a MultilingualBERT (mBERT) model. To account for the lack of a pre-trained BERT model for Arabic, the authors trained four different Arabic BERT models from scratch and made them available to the public.

The primary model used in this study consists of two parts. The first part involved passing the text through 12 self-attention layers of the BERT Base model to obtain contextualized vector representations. The second part involved using a CNN as a classifier. The model was trained for 10 epochs with a learning rate of 2e-5, and the best-performing model based on the macro averaged F1 Score on the development set was saved.

The results of their experiments were compared to other experiments. Initially, a baseline model was created using traditional machine learning methods for text classification. Later, the main model was compared to more advanced techniques. All of these models used the same train/dev/test splits. The results obtained from using only the BERT model were compared to those obtained by combining BERT with CNN, and the latter approach was found to be superior. Furthermore, it was evident that using language-specific pre-trained models provided better results than using Multilingual models. The authors' system has a macro-averaged F1 score of 89.7% in Arabic, a score of 84.3% in Greek, and a score of 81.4% in Turkish, and a score of 85.1% overall.

F. Hate Speech

The prevalence of hateful and abusive content on social media platforms is a significant concern, and it is crucial to address it. Hate speech is considered as an offensive take of expression or opinion towards a certain group or individual. It is a form of harmful message that aims to degrade and ridicule one's ideal or appearance. This is the definition stated in the study conducted by Velankar et al. (2022). In India, Marathi is a well-known regional language used by a significant number of people on social media, but there has been a scarcity of research or progress in the area of detecting hate speech in this language.

The authors presented L3CubeMahaHate, the first and largest publicly available hate speech dataset in Marathi. The dataset was created from X and manually annotated, consisting of over 25,000 posts divided into four categories: hate (HATE), offensive (OFFN), profane (PRFN), and not (NOT), and a binary version (HOF for hate and NOT for not) with over 37,500 samples. They also presented baseline results on 2-class as well as 4-class classification using deep learning models based on CNN, LSTM, BiLSTM, and and transformer-based BERT models, such as IndicBERT, Multilingual BERT (mBERT), and RoBERTa, and monolingual Marathi BERT models, such as MahaBERT, MahaALBERT, and MahaRoBERTa.

For the CNN and LSTM models, the authors employed random and fast text initialization for the word embeddings. The pre-trained embeddings were used in both trainable and non-trainable modes, with the former allowing the embedding layer to adapt to the training data and the latter preventing it from being updated during training. The

study also employed pre-trained language models such as IndicBERT, mBERT, XLM-RoBERTa, and some custom BERT models like MahaALBERT, MahaBERT, and MahaRoBERTa to achieve their results.

The non-trainable fast text mode was the best configuration for both binary and 4-class results in CNN and LSTM models. Monolingual Marathi BERT models outperformed multilingual BERT models such as IndicBERT, mBERT, and XLM-RoBERTa. In fact, the non-trainable fast text mode for CNN and LSTM based models was competitive with the BERT models and even surpassed IndicBERT in both classes. The MahaBERT model provides the best binary classification results with 91% accuracy rate, while MahaRoBERTa gives the best 4-class accuracy of 80%. The study concluded that monolingual models outperformed multilingual models based on the results obtained. The MahaBERT model was found to provide the best results on the L3Cube-MahaHate Corpus.

Posts from a hate speech dataset by Waseem (2016) were classified by different approaches on CNN deep learning models by Gambäck and Sikdar (2017). The study presented a deep learning system for detecting hate speech on X. The system categorizes posts into four groups: racism, sexism, both racism and sexism, and non-hate speech. The system used four CNN models based on different feature embeddings: character 4-grams, word vectors created by word2vec, randomly generated word vectors, and a combination of word vectors and character n-grams.

Each post in the English X hate speech dataset was annotated by one Expert annotator and three Amateur annotators. Waseem defined the "Expert" annotators as

those with theoretical and applied knowledge of hate speech, recruited from feminist and antiracism activists. The "Amateur" annotations were obtained through crowdsourcing on the CrowdFlower platform. The study combined the annotated tags for each post based on majority voting, with the Expert annotators given double unit votes and each of the Amateur annotators given a single unit vote.

The results were compared to the Logistic Regression (LogReg) model used by Waseem and Hovy (2016). The first CNN model used random word vectors and showed significant improvements in precision over the LogReg model, but lower recall. The second model used word2vec embeddings and improved recall by 7.3% compared to the random vector model, achieving an F-score of 78.29%. The third and fourth models used character n-grams in addition to word embeddings. The third model used only character n-grams as feature embeddings, while the fourth model combined word2vec embeddings and character n-grams. The fourth model achieved the best precision, but the word2vec model without character n-grams had the best overall performance with precision, recall, and F-score values of 85.66%, 72.14%, and 78.29%, respectively. The CNN models outperformed the LogReg model in terms of precision and F1 score, while the LogReg model had better recall than the neural network models.

A local study was conducted to analyze the composition of hate speech in posts with topics on the 2016 Philippine Presidential Elections (Solitana & Cheng, 2021). To achieve this, the authors used word bubbles to identify similarities and differences between hate and non-hate expressions, and Principal Component Analysis (PCA) with K-means clustering to confirm the clustering of hate targets. The results of the study showed that there is a 55% difference in terms used between hate and non-hate posts.

However, the clustering of hate targets based on annotation did not match the groupings generated in terms of nearest neighbors. The study also found that there was a significant difference in lexical diversity between hate classes and targets, indicating that it could potentially be used as a feature to improve hate speech classification.

G. Synthesis of Related Literatures and Studies

Table 2.1. Synthesis Table

Title	Author/s	Approach	Conclusion	Findings
Natural Language Processing				
Feature-Based Subjectivity Classification Filipino Text	Regalado and Cheng (2012)	The study gives emphasis on a feature-based approach for subjectivity classification in Filipino Language. Classifiers that were used in the study are Naïve Bayes, Bagging, Multilayer Perceptron, and Random Forest Tree. Features such as part-of-speech (POS), subjectivity, term presence, and frequency information were part of the classification.	Bagging classifier showed an outstanding result of 64.7406% accuracy. This is evident in the first 10-fold cross validation experiment where only opinionated terms during the term frequency and presence feature extraction were used. Random Forest Tree classifier also tied with Bagging classifier in the other two experiments—both having a	Experiment 1 Using Opinionated Keyword - Bagging Classifier performance with POS features + term presence: 64.5% (Recall), 64.4% (Precision), 64.3% (F-Measure), and 64.7% (accuracy).

		Through crowd-sourcing, every sentence in the editorial was tagged as either objective or subjective.	classification accuracy of 63.2941%.	
Sentiment Analysis				
Utilizing Hashtags for Sentiment Analysis of Tweets in The Political Domain	Alfina, etl al. (2017)	The study made use of hashtags as a feature in classification. There is automatic annotation of dataset using the number of positive and negative hashtags as basis (SentiHT feature). The study also made a comparison of classification accuracy using four different algorithms: Naïve Bayes (NB), Support Vector Machine (SVM), Logistic Regression (LR) and Random Forest (RF).	The SentiHT feature as well as the automatically labeled dataset in sentiment classification boasted an accuracy of more than 95%. Whether paired with NB, SVM, or LR algorithms, SentiHT performed better than the unigram feature, which is not the case when RF algorithm is used. The study was able to demonstrate both highly accurate sentiment polarity classification and affordable training dataset annotation due to the	SentiHT feature contributed the most in determining sentiment polarity. Using the SentiHT feature with NB was recommended in terms of the computing time to build the model, with the computing time of 1.97 seconds.

			availability of hashtags.	
Sentiment Analysis of Tweets Using Supervised Learning Algorithms	Mehta, et al. (2020)	The study conducted Sentiment Analysis in order to classify posts through the use of nine different machine learning models namely, Naive Bayes, Support Vector Machine, Decision Tree, Max Entropy, XGBoost, Random Forest, Convolutional Neural Network, Long-short Term Memory, and Multilayer Perceptron. The authors made use of regular expressions and the NLTK library in Python to clean the data during preprocessing.	It was concluded that out of all of the proposed machine learning models, the CNN model with GloVe embeddings performed the best with a 79% accuracy rate. It was also mentioned that the Support Vector Machine model performed similarly.	It can be observed that once the models were implemented, the CNN model was the best performer when it comes to accuracy with 79% followed by SVM with 78.7% and Naive Bayes with 78.1%. While the worst performing models are Decision Tree with 67.8% and Max Entropy with 70.8%.
Sentiment Analysis on Twitter Using Neural Network: Indonesian Presidential Election 2019 Dataset	Hidayatullah, et al. (2021)	The study highlights the obtaining of the best algorithm for classifying sentiment in X data related to the 2019 Indonesian presidential election using neural network	The three traditional machine learning algorithms were not able to surpass the deep neural network algorithms in terms of accuracy even	Deep learning algorithms accuracy results revealed that BLSTM has the highest accuracy of 84.60% Following this, GRU+LSTM has 84.50% accuracy,

		algorithms. The following deep neural network algorithms were used in training the dataset: Convolutional Neural Network (CNN), Long short-term memory (LSTM), CNN-LSTM, Gated Recurrent Unit (GRU)-LSTM and Bidirectional LSTM (BLSTM). Traditional machine learning algorithms were also used to train the dataset which includes: Support Vector Machine (SVM), Logistic Regression (LR) and Multinomial Naive Bayes (MNB). The accuracy of these three algorithms was compared by applying Term Frequency-Inversed Document Frequency (TF-IDF) as a feature and not applying it.	with the use of TF-IDF. BLSTM has an accuracy of 84.60 which makes it the best among the deep learning algorithms.	CNN+LSTM with 84.30%, LSTM with 84.20%, and CNN with 84.05%.
Twitter sentiment analysis using	Roy & Ojha (2020)	The study made a comparison of deep learning	In terms of accuracy, BERT outperformed	BERT model performance: 0.64 (Recall),

deep learning models		models' performance, particularly BERT, attention based Bidirectional LSTM and Convolutional Neural Network (CNN). The SemEval-2016 X dataset was used to train these models with fine-tuned embeddings.	the other models with 64.1%.	0.65 (Precision), 0.63 (F1 Score), and 64.1% (accuracy). Bi-Attentive LSTM and CNN models showed an accuracy of 60.02% and 59.2%, respectively.
Tagalog Sentiment Analysis Using Deep Learning Approach With Backward Slang Inclusion	Boquiren et al. (2022)	The study utilized Tagalog Sentiment Analysis of posts through deep learning algorithms like LSTM and Bi-LSTM with the inclusion of Backward Slang. Random Search algorithm was used during hyperparameter tuning of the two models. Two versions of train sets were used on each model, one that includes backward slang and one that does not. Validation on four different	Bi-LSTM models performed better than LSTM models. Bi-LSTM's lower values of difference in performance metrics between both versions of the model indicate reduced misclassification of labeled posts. Backward Slang in posts are helpful in classifying the spectrum of opinion and improve sentiment performance in posts, especially in political	Difference between two Bi-LSTM models when it comes to performance metrics: 3.45% in accuracy and AUC, and 3.56% in F1 score. LSTM: 4.28% in accuracy and AUC, and 4.39% in F1 score.

		models was accomplished which is followed by validation through performance metrics.	context.	
Sentiment Analysis of Typhoon Related Tweets using Standard and Bidirectional Recurrent Neural Networks	Imperial et al. (2019)	Sentiment analysis was conducted on posts related to typhoon Yolanda within the period ranging from November 1, 2013, to January 31, 2014, as part of the study. Two variations of Recurrent Neural Networks (RNN) were utilized: standard and bidirectional.	The application of standard and bidirectional RNN algorithms in sentiment analysis was successful, with scores exceeding 80% on all evaluation metrics used for both binary and fine-grained classification.	After training with different hyperparameters, the most optimal models obtained high accuracy rates of 81.79% for fine-grained classification using standard RNN, and 87.69% for binary classification using bidirectional RNN. The results of the study indicate that 51.1% of the posts analyzed conveyed positive sentiments.19.8 % of the posts expressed negative sentiments. The remaining 29% of the posts were neutral.
EmHash:	Kaviani and	The proponents	Hashtags in	The

Hashtag Recommendation using Neural Network based on BERT Embedding	Rahmani (2020)	conducted a BERT word embedding model approach with the inclusion of the recommended novel hashtags using neural networks. The proponents aim to find and analyze course and fine grade semantic similarities through prediction with hashtag clusters in vector form.	BERT word embedding hold significance in semantic similarities. The proponents recommended the use of hashtags in text and sentiment analysis with the note that using a varied number of hashtags may be considered both a strength and weakness to the system.	recommended system EmHash which was used for hashtag in tweets, outperformed the preceding methods in terms of recall performance and achieved higher results in terms of precision. Emhash operates as both a coarse-grained and fine-grained level or semantic similarity. This suggested for groups of hashtags for new tweets and then fine-tuning the suggestions using a similarity threshold of 60%. This can be seen as both a strength and a weakness, as it leads to varying numbers of recommended hashtags for different tweets.
Social and Emotional Correlates of Capitalization on Twitter	Chan et al. (2018)	The study conducted a computation analysis on special	A varietal use of capitalization over text includes *meaningful*	After counting for only meaningfully capitalized tokens instead

		orthographic characteristics in terms of their function and distribution over text. Capitalization in text served as a variety of orthographic characteristics and was used to locate patterns and relation with other demographic factors in context. The data gathered was tokenized and counted for any types of capitalization. Using TweetTokenizer.	capitalization, which is a capitalized text with felt intention and expressive value. Capitalization over text-based communication, when used meaningfully, serves as a cue which holds weight and value in sentiment analysis.	of all capitalized tokens in the transformed text, the results of the ANOVA test showed an increase to the margin of significance for gender and sentiment from **p <0.05 and ** =p <0.01 to p<0.001 and p <0.01 for gender and sentiment respectively.
Convolutional Neural Networks (CNN)				
Comparison of Accuracy between Convolutional Neural Networks and Naïve Bayes Classifiers in Sentiment Analysis on Twitter	Sunarya et al. (2019)	The study made use of the Convolutional Neural Network (CNN) classifier model in the sentiment analysis on English posts related to the "Turkey Crisis 2018" topic. The study compared the resulting CNN	Results of the study showed that the accuracy of CNN classifier model performed a higher rate of 88% than the NBC classifier model which had an accuracy rate of 78%. The study's results indicate that a classifier	The NBC classifier model was not able to deliver better performance with an accuracy of 78% than the CNN classifier with 88% accuracy.

		classifier model with the Naïve Bayes Classifier (NBC) model. The process of sentiment analysis began with data retrieval and then involved data classification using TextBlob to categorize posts into positive, negative, or neutral sentiment. The CNN classifier model was constructed using the Keras Functional API model and had 3 convolutional layers, with each layer utilizing 100 kernel filters.	model utilizing a deep learning algorithm provides more accurate sentiment analysis than the Naïve Bayes classifier model.	
A sentiment analysis study for Twitter using the various model of convolutional neural network	Alim (2021)	The study aimed to improve the accuracy of sentiment analysis using deep learning with the CNN algorithm. The Indonesian-Sentiment-Analysis-Dataset was used as well as the Word2Vec model for Indonesian as a word vector	CNN models were proved to perform better compared to SVM, KNN, and SGD in terms of accuracy for sentiment analysis of Indonesian posts. The best-performing CNN model is CNN 12 with 81.4% accuracy,	The best CNN model was CNN 12 with 81.4% accuracy, 81.1% precision and recall, and 81.0% F1 score. Machine learning algorithms' classification accuracy: SGD (62.1%), SVM (61.4%), and KNN (52.3%).

		representation. A variety of CNN models were developed with parameter configurations. The CNN models were compared to machine learning algorithms such as SVM, KNN, and SGD to evaluate their accuracy.	and the experiments showed that the number of convolutional layers, number of filters, and filter size are important parameters for improving the performance of the CNN models. The study also confirmed that deep learning approaches such as CNN can improve the accuracy of sentiment analysis compared to traditional machine learning algorithms.	
Bidirectional Encoder Representations from Transformers (BERT)				
Emotion and sentiment analysis of tweets using BERT	Chiorrini et al. (2021)	Sentiment analysis and emotion recognition using BERT models. For these two tasks, there are two distinct classifiers, uncased BERT-base and cased BERT-base, wherein the	An accuracy of 92% for sentiment and 90% for emotion emotion recognition were obtained which proved that BERT models have exceptional language modeling power.	Emotion recognition: Uncased BERT has an accuracy and F1 score of 89%, while cased BERT has 90% accuracy and 91% F1 score. Sentiment analysis:

		performance of the resulting models is evaluated.		The accuracy and F1 score of the uncased and cased BERT models are comparable, both achieving a score of 92%.
A BERT Framework to Sentiment Analysis of Tweets	Bello, et al. (2023)	Text classification for natural language processing using BERT with other variants including BiLSTM, CNN, and RNN. A comparison of BERT and Word2Vec when combined with the deep learning classifiers was conducted.	In terms of accuracy, recall, and precision, BERT in combination with CNN, RNN, and BiLSTM, performed better when compared to the results of aforementioned models used alone or with the use of Word2Vec.	BERT-CNN, BERT-RNN, and BERT-BiLSTM achieved an accuracy of 93% and F-measure of 95%. On the other hand Word2Vec-CNN , Word2Vec-RNN , and Word2Vec-BiLSTM achieved an accuracy of 57%, 48%, and 55%, respectively, while 64%, 56%, and 63% for the F-measure.
Classification of Fire Related Tweets on Twitter Using Bidirectional Encoder Representations from Transformers (BERT)	Mingua et al. (2021)	The study gathered a dataset consisting of 2,081 Filipino fire related posts for training and testing. The study also made use of different pre-trained BERT models to	In terms of accuracy, the test performance of the BERT Base Uncased WWM model has shown to outperform the other BERT models with 87.50%, proving that pre-trained BERT models	The BERT Base Uncased WWM model produced the best accuracy of 87.50% in test performance and a loss of 6% in training while the BERT Base Cased WWM model produced the lowest test

		classify Filipino posts with context of fire related content.	are effective in classifying posts in fire related context.	accuracy of 76.34% and train loss of 20%.
Context-Based Profanity Detection and Censorship Using Bidirectional Encoder Representations from Transformers	Galinato et al. (2023)	The authors of the study gathered around 14,000 Tagalog posts using SNScrape to particularly select those containing Tagalog profanity. They created the dataset Tagalog X Profanity (TTP). Based on the pre-existing Tagalog BERT model, they proposed an improved model to detect and classify the contexts and usage of Tagalog profanities in posts. This also includes censorship of Tagalog texts.	BERT models perform well in classifying Tagalog context-based profanities in posts under the right annotations (TTP dataset).	The developed BERT model specifically gained a total of 86% accuracy performance across both abusive and non abusive Posts, while the dataset used had a 96% overall inter-annotator agreement.
How Different Text-Preprocessing Techniques using the Bert Model Affect the Gender Profiling of Authors	Alzahrani & Jololian (2021)	The authors of the study proposed different preprocessing techniques to be used in a Bert Model that predicts the	The authors of the study concluded that the BERT model with no processing techniques used outperformed all the other models	The major finding in this study is the fact that BERT models tend to perform better with little to no preprocessing applied to the

		gender of authors through the input text. These preprocessing techniques are grouped into cases which are: Case 1 that removes mentions, retweet tags, and hashtags; Case 2 that adds the removal of URLs; Case 3 that adds the removal of punctuations; Case 4 that adds the removal of stop word; and Case 5 which is the absence of preprocessing.	tested.	text. This is due to the fact that pretrained BERT models perform better on large text where there are more tokens to learn from, at the same time the authors noted that transfer-learning techniques are feature independent and are capable of understanding natural language.
BERT-CNN				
A Commodity Review Sentiment Analysis Based on BERT-CNN Model	Dong et al. (2020)	Sentiment analysis of commodity reviews was conducted using a BERT-CNN model. The BERT-CNN model was compared with BERT model and CNN model.	The BERT-CNN model performed significantly better than the BERT and CNN models, with an F1 value improvement of approximately 14.4% and 17.4%, respectively. The BERT-CNN model also showed good generalization ability.	BERT-CNN model performance on training dataset: 85.6% (Precision), 84.7% (Recall), and 85.1% (F1 value). BERT-CNN model performance on test dataset: 85.4% (Precision), 84.3% (Recall), and 84.3% (F1 value).

KUISAIL at SemEval-2020 Task 12: BERT-CNN for Offensive Speech Identification in Social Media	Safaya et al., 2019	Pre-trained BERT and CNN models were combined for text analysis. Posts have been labeled as either Offensive (positive) or Non-offensive (negative). The authors used four different pre-trained BERT models including three language-specific models and a MultilingualBERT model. Evaluation metric used was macro-averaged F1-Score where their experiments were compared with other experiments.	The study demonstrated that combining BERT and CNN models was more effective compared to using each model individually, highlighting the significance of pre-trained language models for downstream tasks like detecting offensive language.	The BERT-CNN model has a macro-averaged F1 score of 89.7% in Arabic, 84.3% in Greek, 81.4% in Turkish, and 85.1% overall.
Hate Speech				
L3Cube-MahaHate: A Tweet-based Marathi Hate Speech Detection Dataset and BERT models	Velankar et al. (2022)	The authors of the study created a dataset consisting of posts within the context and language of Marathi. They developed and trained different deep	The transformer BERT models proved to provide the best outputs in detecting hate speech in binary classification as well as 4-class classifications. Among the	Model performance in 2-Class Accuracy: CNN (Random) with 88%, LSTM (Non-Trainable) with 87%, BiLSTM (Non-Trainable) with 87%, MahaBERT with

		learning models and both multilingual and monolingual transformer-based BERT models which can identify hate speech in categories of not (*NOT)*, profane (*PRFN*), offensive (*OFFN),* and hateful (*HATE*) for 4-class classification, and hate (HOF) and not (NOT) for binary classification.	developed deep learning based models, the MahaBERT model performed best in binary classification, while the MahaRoBERTa outperformed all BERT models in terms of 4-class classification. For other deep learning models such as CNN or LSTM, non-trainable fast text is found to perform better than trainable fast text for both binary and 4-class classification than other BERT models.	91%. Model performance in 4-Class Accuracy: CNN (Non-Trainable) with 75%, LSTM (Non-Trainable) with 75%, BiLSTM (Non-Trainable) with 76%, MahaRoBERTa with 80%.
Using Convolutional Neural Networks to Classify Hate-Speech	Gambäck and Sikdar (2017)	The authors of the study created a hate speech identifier in English posts using two or more models of CNN.	The developed CNN model performed better in terms of precision and F1 score compared to other models such as LogReg.	The version of the CNN model which utilized word2vec outperformed all other CNN models with an F-Score of 78.29%, Precision of 85.66% and Recall of 72.14%. It is also found that all developed versions of the CNN models managed to perform better in Precision and F1 score than

				LogReg.
Analyses of Hate and Non-Hate Expressions during Election using NLP	Solitana and Cheng (2021)	Word bubbles and Principal Component Analysis (PCA) with K-means clustering were the main techniques used to analyze the composition of hate speech.	There is a significant difference in terms used between hate and non-hate posts. Clustering of hate targets based on annotation did not match the groupings generated in terms of nearest neighbors, Lexical diversity could potentially be used as a feature to improve hate speech classification	There is a 55% difference in terms used between hate and non-hate posts, and significant difference in lexical diversity between hate classes and targets.

Chapter III

Research Design and Methodology

This chapter of the study discussed the methods and design applied to the model architecture of the system. Methods to data gathering, preprocessing, modeling and evaluation of prediction were elaborated with necessary illustrations. This chapter elaborated the process of developing the improved model in detecting election-related hate speech in posts.

A. Hypotheses and Assumptions

The researchers of the study have deduced the following hypotheses for this study:

H_o: There is no significant difference in the performance between the BERT-CNN model and *fastText* CNN in the sentiment analysis of Philippine election-related posts into hate or non-hate classification.

H_a: There is a significant improvement in the performance between the BERT-CNN model and *fastText* CNN in the sentiment analysis of Philippine election-related posts into hate or non-hate classification.

Assumptions

The following statements were assumed to be true in this study:

- The labeled data used in the sentiment analysis was correct and accurate.
- The dataset contained sufficient data for the system to output correct predictions.

- The size input for training and testing of data was sufficient to form an accurate analysis of the BERT-CNN model's performance.

B. Research Method

The main goal of this study was to implement a model which detected and identified hate speech in Philippine election-related posts using the BERT-CNN model to improve the performance of the original *fastText* CNN model of the study conducted by Argañosa et al. (2022). The study employed a comparative research method to evaluate the performance of the two systems. This approach enabled the researchers to compare the performance of the systems and determine if there was a significant improvement due to the combination of BERT and CNN, as well as the necessary modifications made such as the inclusion of hashtags as well as ALL-CAPS and a combination of both hashtags and ALL-CAPS features.

Data gathering was the initial step to the process of developing the proposed model. The researchers of the study gathered the data through the labeled dataset of posts collected from the study by Argañosa et. al. (2022).

Subsequently, the gathered data were then manually labeled based on their sentiment and classified whether they are posts of positive, negative, or neutral sentiment for multi label classification, and hate or non-hate for binary classification. The dataset underwent extensive preprocessing steps for analysis. These steps included removing account mentions, URLs, special characters, and candidate names. English and Filipino stopwords were retained, and hashtags as well as ALL-CAPS were incorporated. Variations were introduced to address the impact of hashtags, ALL-CAPS, or both on the

BERT-CNN model. Hashtags were excluded for the base BERT-CNN and BERT-CNN with ALL-CAPS but retained for other models. The normalization process preserved ALL-CAPS casing for relevant models. Finally, the textual data was tokenized for effective vectorization.

The modeling process involved the preprocessed dataset which were used to train a BERT model paired with CNN as according to the related literature, these models combined performed well when it comes to classifying posts. Lastly, the performance evaluation determined the quality of the output along with the performance of the BERT-CNN model.

C. Research Design

The presented section of the study focused on the model design to further discuss the collection method, analysis, and other considerations for the data. This study was divided into the following steps: Information Gathering, Research Analysis, Designing the Procedures and Algorithms, Implementation of the Model, and Testing and Debugging.

a. Information Gathering

The study was built upon the previous study titled "Hate Speech in Filipino Election-Related Posts: A Sentiment Analysis Using Convolutional Neural Networks" by Argañosa et al. (2022) through the use of an improved system architecture. For the dataset, the researchers utilized the labeled posts collected from the base study by Argañosa et. al. (2022) related to the 2022 Philippine elections and pertained to the presidential candidates.

b. Research Analysis

Information gathered from the previous step was analyzed in this second step. The researchers of the study employed BERT-CNN in conducting sentiment analysis with regards to the posts gathered from X as the main platform source. It was also through the recommendations made by the authors (Argañosa et al., 2022) that the researchers decided to use CNN and combined it with BERT to improve the sentiment analysis results. The posts gathered retained hashtags as per recommendation in the base study that may provide a better description for the dataset and potentially improve the sentiment analysis results. In addition to that, ALL-CAPS were also retained. For sentiment analysis, the preprocessed posts were classified into classes: positive, negative, and neutral for multi label classification, and hate or non-hate for binary classification. With the employment of BERT-CNN and utilization of hashtags and ALL-CAPS, the researchers aimed to improve the sentiment analysis of election-related posts into hate or non-hate classification.

c. Designing the Procedures and Algorithms

The proposed study focused on improving the original algorithm through the use of the BERT-CNN algorithm. The design and procedures of the study were geared towards increasing the effectiveness of the sentiment analysis on Philippine election-related posts. The same preprocessing methods used in the base study were utilized in this study as well. The only difference was the inclusion of stopwords, hashtags, and ALL-CAPS, which improved the accuracy and effectiveness of the sentiment analysis. The researchers carefully designed

and implemented their procedures and algorithms to ensure that the study achieved its objective of improving the existing algorithm using the BERT-CNN algorithm and produce more accurate and insightful results for sentiment analysis of Philippine election-related posts with interest in the campaign's presidential candidates.

d. Implementation of the Model

With alignment to the objectives of the study, the researchers performed the necessary procedures in implementing the modified models based on the set framework. The programming for the model was carried out by the researchers where they applied the concepts, procedures, and algorithms designed in the previous step. To accomplish this, the researchers used the Python programming language as it provided a range of natural language processing extensions that are available within its libraries. By doing so, the researchers leveraged the features of Python to process the data effectively and efficiently, thus enabled them to generate accurate and reliable results for sentiment analysis of Philippine election-related posts.

e. Testing and Debugging

This step of the study's research design utilized testing and debugging techniques to investigate issues within the design and implementation of the model. Each line of code and parameters was carefully observed during implementation and ensured that each function in the preprocessing and modeling process would work as intended. Multiple executions of the model were done to determine any gaps in performance. For the algorithms, evaluation and dataframe

methods were used to find any errors and inconsistencies that may appear in execution.

f. Evaluation

The BERT-CNN model developed in this study was assessed based on the accuracy, precision, recall, and F1 score. The performance of the model was then compared to the *fastText* CNN model of Argañosa et al. (2020) in terms of their ability to conduct sentiment analysis on Philippine election-related posts. In addition, the BERT-CNN model in combination with features was compared with the *fastText* CNN model. The said combinations were the following: (1) BERT-CNN with Hashtags, (2) BERT-CNN with ALL-CAPS, and (3) BERT-CNN with Hashtags and ALL-CAPS. Lastly, the researchers derived the insights from the analyzed text data to gain a better understanding of the sentiment expressed in the posts.

D. Specifications of Methodology

This section of the study explained the tools which were used in the gathering data, preprocessing, modeling, and evaluation process as shown in the following subsections.

a. Research Instrument

The researchers of the study used the following instruments in implementing the machine learning workflow:

- **Hardware**: The preprocessing and modeling steps were implemented on a machine with specifications no lower than 8GB of RAM.
- **Software**: The open source software Jupyter Notebook (anaconda3) along with various imports were employed to implement the design workflow. The programming language Python was used.
- **Peopleware:** The primary contributors of the study were the students who carried it out. Nevertheless, the success of the model was not solely attributed to the students, but to their thesis coordinator and adviser as well. They offered their technical knowledge, skills, experience to support the students' study, recommendations, feedback, and guidance which helped improve the study.

b. Sources of Information

Reputable journals and previous theses were sources of relevant data and information. This study used the dataset by Argañosa et. al. (2022) which played a crucial role in acquiring posts directly relevant to the study's focus. Leveraging this dataset, obtained from reputable sources through a reliable methodology, ensures the reliability and high quality of the data underpinning our study. By building upon the base study, this study ensures a firm grounding in a sound and trustworthy empirical basis, thereby enhancing the credibility of its research outcomes.

E. Sampling and Data Gathering

The following subsections were further discussed in the methods and techniques used in sampling and gathering data for processing.

a. Sampling

The researchers conducted a judgment or purposive sampling technique for this study as the hashtags and threads from the 2022 presidential election posts spanning from October 2021 to May 2022 came from a collection of posts made by various accounts. The data split ratios that were implemented for training and testing of data were 70:30, 80:20, and 90:10 respectively as similar to the base study's hypothesis testing especially for data classification and sentiment analysis.

b. Data Gathering

The researchers mainly gathered the data from the labeled posts collected for the base study by Argañosa et. al. (2022) which provided posts with complete texts related to the 2022 Philippine elections with interest of the presidential candidates. The data gathered were posts between October 8, 2021 to May 7, 2022. As stated, the study collected all training and testing posts, including those with hashtags, as they served as a basis and a significant point in building the improved model with respect to the base study.

F. Statistical Treatment

To evaluate the results of the study whether the objectives have been achieved, the researchers employed a set of evaluation and performance metrics to investigate the

effectiveness of using BERT combined with CNN as compared to the use of CNN models with *fastText* in detecting hate speech in Philippine election-related posts.

a. Confusion Matrix

The confusion matrix is a measurement of performance in classification problems for both binary and multi label types. This also served as a measurement for metrics in Recall, Precision, F1 Score, and Accuracy. For binary classification, the predicted values were labeled based on the classification from the confusion matrix: True Positive *(TP),* False Positive *(FP),* True Negative *(TN), and* False Negative *(FN).*

Table 3.1. Confusion Matrix for Binary Classification

		Predicted	
		Hate Speech	**Non-Hate Speech**
Actual	**Hate Speech**	True Positive (TP)	False Negative (FN)
	Non-Hate Speech	False Positive (FP)	True Negative (TN)

For multi label classification, True Positive *(TP)* is the number of correct predictions for positive instances. False Positive1 *(FP1)* and False Positive2 *(FP2)* are the number of incorrect predictions, where negative or neutral instances are predicted as positive, respectively. True Neutral *(TNt)* is the number of predictions which is true for neutral actual data. False Neutral1 *(FNt1)* and False Neutral2 *(FNt2)* are the number of predictions that are wrong, where positive or negative instances are predicted as neutral, respectively. True Negative *(TNg)* is the number of correct predictions for negative actual data. False Negative1 *(FNg1)* and False Negative2 *(FNg2)* are the number of incorrect predictions, where positive or neutral instances are predicted as negative, respectively.

Table 3.2. Confusion Matrix for Multi Label Classification

		Predicted		
		Positive	**Negative**	**Neutral**
Actual	**Positive**	True Positive (TP)	False Negative1 (FNg1)	False Neutral1 (FNt1)
	Negative	False Positive1 (FP1)	True Negative (TNg)	False Neutral2 (FNt2)
	Neutral	False Positive2 (FP2)	False Negative2 (FNg2)	True Neutral (TNt)

b. Accuracy

Accuracy is a metric which measures the closeness of a prediction to its true value.

i. Binary Classification

$$Accuracy_B = \frac{TN + TP}{TN + FN + TP + FP} \quad (3.1)$$

ii. Multi label Classification

$$Accuracy_M = \frac{TP + TNg + FNt}{TP + FNg1 + \ldots + FNg2 + TNt} \quad (3.2)$$

c. Precision

Precision is the metric of measuring the closeness of correct values to each other. It provided a good description of noises and potential errors for the model's output.

i. Binary Classification

$$Precision_B = \frac{TP}{TP + FP} \quad (3.3)$$

ii. Multi label Classification

$$Precision_{M(+)^*} = \frac{TP}{TP + FP1 + FP2} \quad (3.4)$$

$$Precision_{M(-)^*} = \frac{TNg}{FNg1+TNg+FNg2}$$

(3.5)

$$Precision_{M(0)^*} = \frac{TNt}{FNt1+FNt2+TNt}$$

(3.6)

**(+) Positive; (-) Negative; (0) Neutral*

d. Recall

Recall is the metric of measuring the amount of correctly predicted values based on all of the previously learned predictions.

i. Binary Classification

$$Recall_B = \frac{TP}{TP+FN}$$

(3.7)

ii. Multi label Classification

$$Recall_{M(+)} = \frac{TP}{TP+FNg1+FNt1}$$

(3.8)

$$Recall_{M(-)} = \frac{TNg}{FP1+TNg+FNt2}$$

(3.9)

$$Recall_{M(0)} = \frac{TNt}{FP2+FNg2+TNt}$$

(3.10)

e. F1 Score

The F1 Score is an evaluation metric that incorporates both precision and recall metrics to provide a single performance measure. It can be thought of as the harmonic mean of precision and recall.

i. Binary Classification

$$F1\ Score_{B} = 2 \cdot \frac{Precision_{B} \cdot Recall_{B}}{Precision_{B} + Recall_{B}} \quad (3.11)$$

ii. Multi label Classification

$$F1\ Score_{M(+)} = 2 \cdot \frac{Precision_{M(+)} \cdot Recall_{M(+)}}{Precision_{M(+)} + Recall_{M(+)}} \quad (3.12)$$

$$F1\ Score_{M(-)} = 2 \cdot \frac{Precision_{M(-)} \cdot Recall_{M(-)}}{Precision_{M(-)} + Recall_{M(-)}} \quad (3.13)$$

$$F1\ Score_{M(0)} = 2 \cdot \frac{Precision_{M(0)} \cdot Recall_{M(0)}}{Precision_{M(0)} + Recall_{M(0)}} \quad (3.14)$$

f. Macro Average (Multi Label Classification)

Let *SumTP, SumFP, SumTN, SumFN* be the sum of all True Positives, True Negatives, False Positives, and False Negatives respectively.

i. Precision

$$Precision\_MacroAvg = \frac{(Prec_1 + Prec_2 + \ldots + Prec_n)}{n}$$

(3.15)

ii. Recall

$$Recall\_MacroAvg = \frac{(Recall_1 + Recall_2 + \ldots + Recall_n)}{n}$$

(3.16)

iii. F1 Score

$$F1\ Score\_MacroAvg = \frac{(F1\ Score_1 + F1\ Score_2 + \ldots + F1\ Score_n)}{n}$$

(3.17)

g. McNemar's Test with Holm-Bonferroni Correction

The utilization of statistical hypothesis testing entails an examination of likelihood scores between samples, assuming they are drawn from the same distribution (Brownlee, 2018). The application of statistical hypothesis testing, such as McNemar's 2x2 Validation Testing, aids in assessing the null hypothesis assumption regarding whether the disparity between two models should be accepted or rejected. The values from Table 3.3 were employed to visually compare the two models based on the count of accurate and inaccurate predictive values.

Table 3.3. McNemar's Test Table

		$Model\ A$	
		Correct (+)	Wrong (-)
$Model\ B$	Correct (+)	a	b
	Wrong (-)	c	d

The formula from equation 3.11 was used to calculate the paired data where *b* and *c* are discordant frequencies.

$$z^2 = \frac{(|b-c| - 1)^2}{b+c} \quad (3.18)$$

Chapter IV

Presentation and Analysis of Data

This chapter of the study presented the results and analysis of data processed by the researchers' built System Architecture, Module Tests, Sample System Tests and Results in detecting hate speech from 2021 to 2022 election related posts.

A. System Architecture

In relation to the methods of the study, the system architecture visualized the flow of process in turning the data into binary labeled or and multi labeled posts with context to the 2022 Philippine presidential elections. Each phase in the system architecture aimed to address the objectives of the study. The group segmented the system architecture into five (5) modules namely, the Preprocessing, Splitting, BERT Word Embeddings, Binary and Multi Label CNN, and Evaluation.

Figure 4.1 below organizes the modules in alignment with their respective processes. Each stage within the modules represents a step in which data undergoes transformation, contributing to the construction of the model. The processes encapsulated with dashed lines represented the feature inclusion of the study, specifically the implementation of test cases into the system's model and sentiment analysis such as the inclusion of hashtags and/or ALL-CAPS. The processes enclosed in the slanted boxes indicated the expected data flow, whether it be input or output for the system. Further discussions of the system architecture were explained in detail to its components' processes, methods, and sub-components utilized.

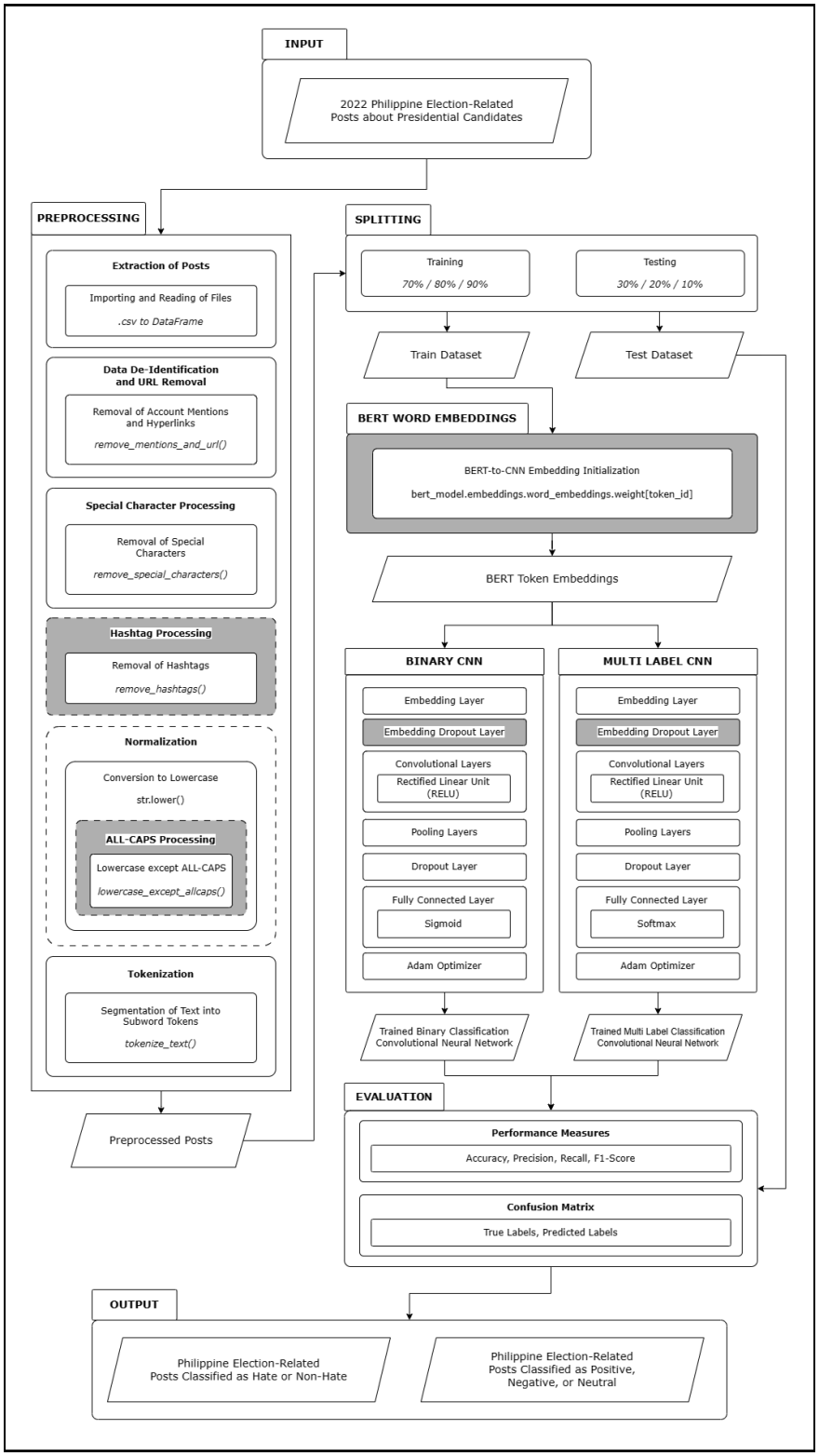


Figure 4.1. System Architecture

The input data of the system architecture were extracted from the labeled dataset of the base study Arganosa et. al. (2022) containing the gathered data of election-related posts from the X API during the 2022 Philippine elections. The validated dataset of the base study was the key comparison to the four (4) types of dataset that resulted from the implemented BERT-CNN models compared to *fastText* CNN: (1) BERT-CNN, (2) BERT-CNN with Hashtags, (3) BERT-CNN with ALL-CAPS, and (4) BERT-CNN with Hashtags and ALL-CAPS. The system architecture is made up of five (5) major modules namely, the Preprocessing, Splitting, BERT Word Embeddings, Binary and Multi Label CNN, and Evaluation. The Preprocessing is composed of eight (6) submodules: Extraction of Posts, Data De-Identification and URL Removal, Special Character Processing, Tokenization as well as the innovative changes that were added to the system namely, the **Normalization with ALL-CAPS Processing**, and **Hashtag Processing**. The next module, Splitting, segmented the preprocessed data into training at 70/80/90% portion and testing at 30/20/10% portion wherein the train dataset was used for the added **BERT word embeddings** in replacement for the fastText embedding algorithm. The BERT token embeddings was passed to two (2) types of CNN namely, Binary CNN and Multi Label CNN with the additional layer of **Embedding Output** for the BERT algorithm. The Evaluation module then tested the trained Binary CNN and Multi label CNN which resulted in two types of outputs: Classified posts as either *Hate*, *Non-Hate* or *Positive, Neutral, Negative*.

B. Description of Modules and Interfaces

This section of the chapter discusses the individual modules within the system architecture of the study. The system architecture is composed of the following modules: (1) Preprocessing, (2) Splitting, (3) BERT Word Embeddings, (4) Binary CNN and Multi Label CNN, and (5) Evaluation which were systematically implemented for detecting hate speech in presidential election-related posts.

B.1. Preprocessing

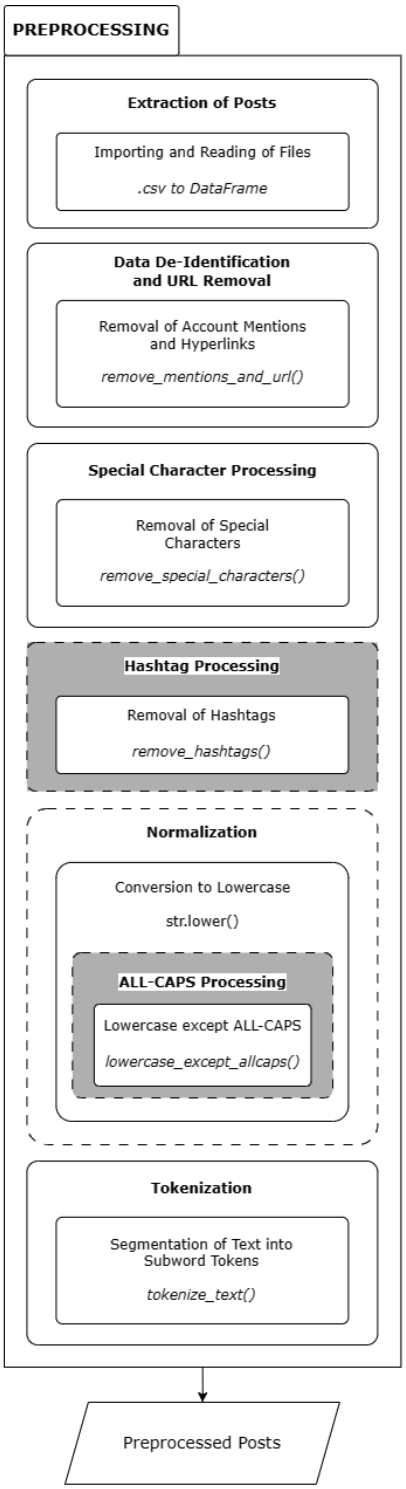


Figure 4.2. Preprocessing Module

After extracting posts, the preprocessing module was executed to organize, clean, and transform the posts into readable and modifiable data. This was done through eight (6) submodules: Extraction of Posts, Data De-Identification and URL Removal, Special Character Processing, Hashtag Processing, Normalization with inclusion to ALL-CAPS Processing, and Tokenization.

The initial step involved the extraction of data, transforming user posts into CSV and dataframe values for improved readability. Once the data was extracted, the preprocessing stage required several steps of cleaning to ensure that the model was to be fed with accurate and complete information. The data was then checked for any account mentions, URLs, special characters including emojis, diacritics, and numerics. In contrast to the methodology employed by the base study, where English and Filipino stopwords were eliminated, this study chose to retain them. The utilization of hashtags and ALL-CAPS were also one of the few changes the researchers revised from the base study's system. This created more than one dataset, four for each type of classification: dataset without Hashtags and ALL-CAPS, dataset with Hashtags, dataset with ALL-CAPS, and dataset with Hashtags and ALL-CAPS. For hashtag processing, hashtags were removed in the base BERT-CNN and BERT-CNN with ALL-CAPS models but retained for others utilizing hashtags. Concerning ALL-CAPS processing, this step occurred during normalization. Although every text was converted to lowercase, the casing for ALL-CAPS was preserved for models that utilized them. Finally, the textual data underwent tokenization using BERT tokenizer to facilitate subsequent vectorization processes.

B.2. Splitting

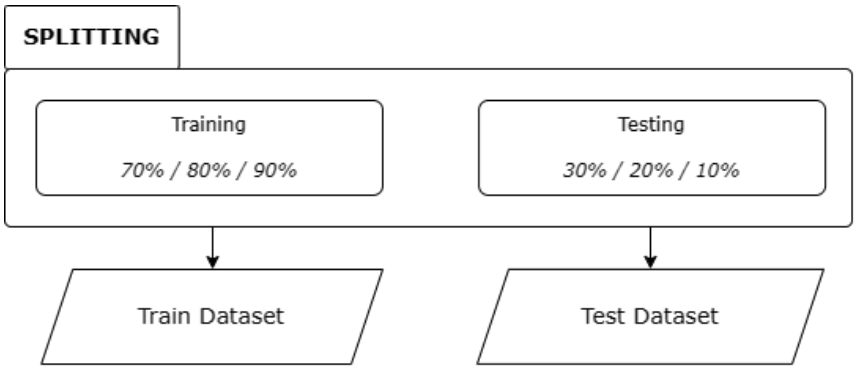


Figure 4.3. Splitting Module

Following tokenization, the preprocessed data was split into training and testing datasets. During training, accurate classification across hate and non-hate labels, as well as positive, neutral, and negative sentiments took place. Adjustments to the training set's parameters were implemented, enhancing the model's capacity to discern underlying patterns and critical correlations within the data.

Once the training phase was complete, attention shifted to the testing set, comprising new, unseen data for the model to classify. The model's performance was then assessed based on what it had learned during training. The partitioning of the data into training and testing sets adhered to a ratio scheme: 70%-30%, 80%-20%, and 90%-10%, summing up to 100%. This strategic partitioning allowed for a robust evaluation of the model's generalization capabilities across various proportions of training and testing data.

B.3. BERT Word Embeddings

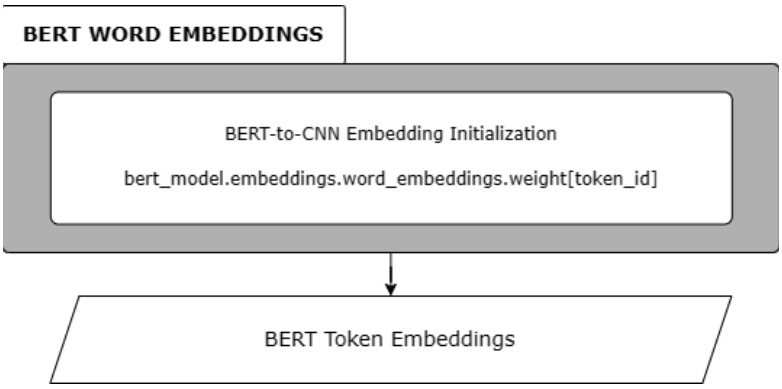


Figure 4.4. Pre Trained BERT Word Embeddings Module

The BERT word embeddings module was the next major module after splitting the training dataset. Utilizing the tokenized textual data established during the preprocessing, this module fed the input to the BERT model for word embedding. The generated BERT token embeddings, obtained by extracting token embeddings from the BERT model's embedding layer will be transferred to CNN model's embedding layer. The token embeddings were enriched with contextualized representations of the input. These embeddings became a crucial component in the building process of both the Binary and Multi-Label CNN models. By incorporating the BERT token embeddings, these models were able to leverage the contextual information captured by BERT, thereby enhancing their learning processes and improving their understanding of the intricacies within the data.

B.4. Binary and Multi Label CNN

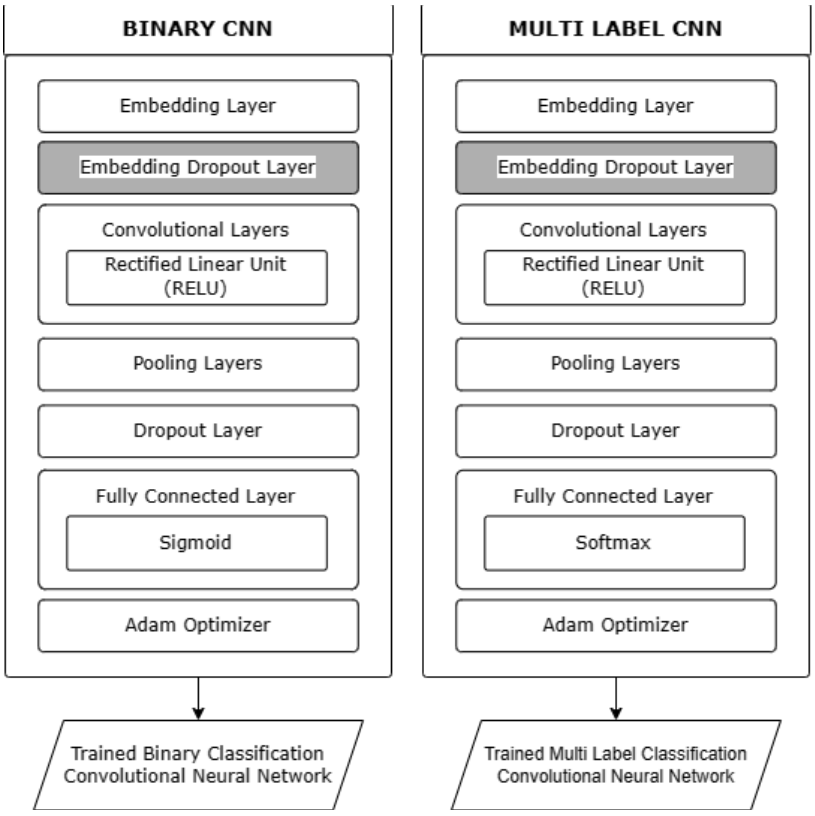


Figure 4.5. Binary and Multi label CNN Module

The data classification in detecting hate speech was segmented into two (2) types: Binary Classification (Hate or Non-Hate) and Multi Label Classification (Positive, Negative, or Neutral). Each labeled data in the training set, after being embedded, was used to train the Binary CNN and the Multi Label CNN models. The two models began with an embedding layer that transformed input features into dense vectors. To curb overfitting, an embedding dropout layer was strategically placed in the architecture. The subsequent convolutional layers, featuring Rectified Linear Unit (RELU) activations, captured hierarchical features in the data. Pooling layers followed suit, reducing spatial

dimensions while retaining essential information. A dropout layer further aided in regularization, preventing co-adaptation of feature detectors. The network diverged at this point based on the task: for binary classification, a fully connected layer with a sigmoid activation produced binary predictions; for multi label classification, a fully connected layer with softmax activation enabled the model to assign probabilities across multiple classes. In both cases, the adam optimizer efficiently updated model parameters during training. The outcome is a trained CNN capable of making predictions tailored to the specific classification task, whether it be binary or multi-label.

B.5. Evaluation

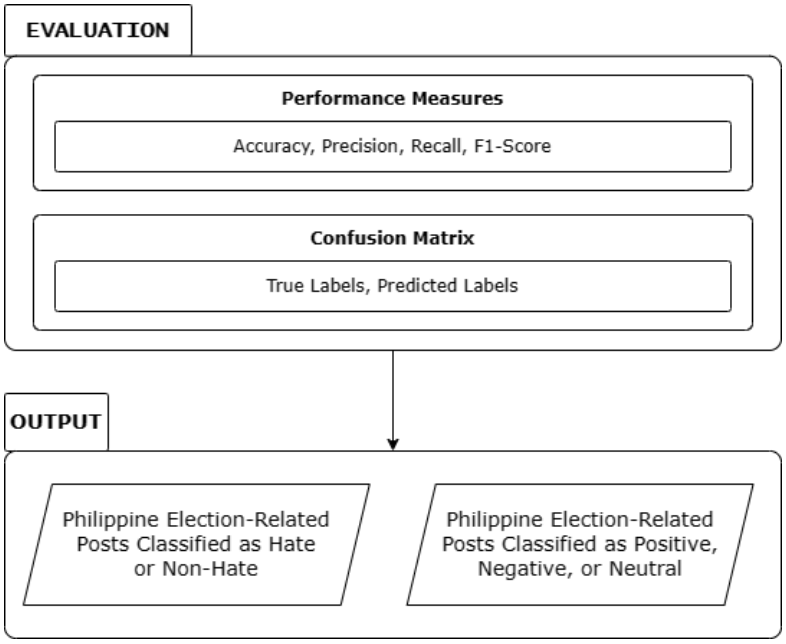


Figure 4.6. Evaluation Module

Once the training portion of the dataset has been processed by BERT to generate contextual embeddings, and this data is used to train the CNN models, the system is then prepared to classify the testing dataset. The anticipated output would manifest in two distinct categories, as depicted in Figure 4.6: Classified Philippine Election-Related Posts labeled as Hate or Non-Hate and Classified Philippine Election-Related Posts labeled as Positive, Negative, or Neutral. To assess the performance of the system in these classifications, key evaluation metrics such as accuracy, precision, recall, and F1-score will be employed. Additionally, a confusion matrix will be generated to provide a detailed breakdown of the model's predictions.

C. Test Cases

This section of the paper succeeds the discussion of the data tested after all the necessary parameters and features were changed to better the model’s accuracy and overall performance in classifying hate speech. There were two (2) subclasses for the model namely the Binary Classification CNN and Multi Label Classification CNN. Each subclass holds four (4) test cases with modified and switched features.

For both Binary and Multi label CNN we have the test cases: a.) BERT-CNN without Hashtags and ALL-CAPS, b.) BERT-CNN with Hashtags, c.) BERT-CNN with ALL-CAPS included, and finally d.) BERT-CNN with Hashtags and ALL-CAPS. The raw data needed to be processed and cleaned before it was used in the BERT-CNN modules.

C.1. BERT-CNN without Hashtags and ALL-CAPS

Table 4.1. Test Case for BERT-CNN without Hashtags and ALL-CAPS

Test Case No. 1
Filename: no_hashtags_allcaps - no_hashtags_allcaps.pdf

Original Tweet	Cleaned Tweet
KULANG ATA SA TULOG SI NORBERTO GONZALES HAHAHAHAHHAHAHAHAHA SINASABI MO DIYAN	kulang ata sa tulog si norberto gonzales hahahahahhahahahaha sinasabi mo diyan
Hirosii's argument screams misogyny not valid information. The thing here is that you can not defend Bongbong Marcos without blabbering unethical arguments lol.	hirosii's argument screams misogyny not valid information. the thing here is that you can not defend bongbong marcos without blabbering unethical arguments lol.
Who is the better vote? Bongbong #marcos, who is funded by the dictators son, or human rights lawyer Roberdo? The answer is obvious. #TheFilipinoVotes #Voting	who is the better vote? bongbong , who is funded by the dictators son, or human rights lawyer roberdo? the answer is obvious.
@skzoowifey hindi siya ang tatay niya bc from what I've learned in my history class, Ferdinand Marcos is smart unlike Bongbong who faked his degree lol. Bongbong is also a tax evader. He can't attend on debates and you know why? He has no balls. Or maybe he's traumatized nung 2016 elections	hindi siya ang tatay niya bc from what i've learned in my history class, ferdinand marcos is smart unlike bongbong who faked his degree lol. bongbong is also a tax evader. he can't attend on debates and you know why? he has no balls. or maybe he's traumatized nung 2016 elections
@Pontifex @LaityFamilyLife Bongbong Marcos,Sara Zimmerman Duterte and their friends from New York will do anything for Bongbong Marcos and Sara Duterte to win Philippine Presidential and Vice-Presidential elections. And then Bongbong Marcos will become a dictator just like his father-so he can do anything.	bongbong marcos,sara zimmerman duterte and their friends from new york will do anything for bongbong marcos and sara duterte to win philippine presidential and vice-presidential elections. and then bongbong marcos will become a dictator just like his father-so he can do anything.
Reference: Bongbong Marcos website and YouTube. Kuya sir, akala ko po nagresearch kayo. #halalanspace	reference: bongbong marcos website and youtube. kuya sir, akala ko po nagresearch kayo.
tipong matatakot ang mga demons lumapit #PasigLabanParaSaTRoPa #PasigLaban (Photo: Team Leni Robredo) https://t.co/3iwaEeUBtd	tipong matatakot ang mga demons lumapit (photo: team leni robredo)
norberto gonzales is a whole-ass boomer, and i'm glad that nobody will ever vote for his fascist ass	norberto gonzales is a whole-ass boomer, and i'm glad that nobody will ever vote for his fascist ass
Goodnight! Sarap makipgbardagulan sa mga trolls at minions hahahaha bukas ulit! Pag bastos na mga bibig auto blocked mala-Ping Lacson lang hahahaha. □□	goodnight! sarap makipgbardagulan sa mga trolls at minions hahahaha bukas ulit! pag bastos na mga bibig auto blocked mala-ping lacson lang hahahaha. □□
@lovsyejisu we're facing an election rn and we want Leni Robredo to win (that's why we're using this pink border) against a son of a dictator.	we're facing an election rn and we want leni robredo to win (that's why we're using this pink border) against a son of a dictator.
Hayop na talunang Dilawang fenk Si Fake Vp Leni Robredo, was pa ngang drug test result until now eh https://t.co/ONtXhMpROQ	hayop na talunang dilawang fenk si fake vp leni robredo, was pa ngang drug test result until now eh
@TitaATIN3 oh yes I am sure. atleast ako hindi stuck sa fake infos na ipinapakalat ng co-supporters mo. Your opinion can be. valid if you are supporting other candidates like Ka Leody, Ping Lacson , Isko Moreno because somehow these other candidates ay may nagawa para sa Pilipinas+	oh yes i am sure. atleast ako hindi stuck sa fake infos na ipinapakalat ng co-supporters mo. your opinion can be. valid if you are supporting other candidates like ka leody, ping lacson , isko moreno because somehow these other candidates ay may nagawa para sa pilipinas+

C.2. BERT-CNN with Hashtags and without ALL-CAPS

Table 4.2. Test Case for BERT-CNN with Hashtags and without ALL-CAPS

Test Case No. 2
Filename: hashtags_no_allcaps - hashtags_no_allcaps.pdf

Original Tweet	Cleaned Tweet
KULANG ATA SA TULOG SI NORBERTO GONZALES HAHAHAHAHHAHAHAHAHA SINASABI MO DIYAN	kulang ata sa tulog si norberto gonzales hahahahahhahahahaha sinasabi mo diyan
Hirosii's argument screams misogyny not valid information. The thing here is that you can not defend Bongbong Marcos without blabbering unethical arguments lol.	hirosii's argument screams misogyny not valid information. the thing here is that you can not defend bongbong marcos without blabbering unethical arguments lol.
Who is the better vote? Bongbong #marcos, who is funded by the dictators son, or human rights lawyer Roberdo? The answer is obvious. #TheFilipinoVotes #Voting	who is the better vote? bongbong #marcos, who is funded by the dictators son, or human rights lawyer roberdo? the answer is obvious. #thefilipinovotes #voting
@skzoowifey hindi siya ang tatay niya bc from what I've learned in my history class, Ferdinand Marcos is smart unlike Bongbong who faked his degree lol. Bongbong is also a tax evader. He can't attend on debates and you know why? He has no balls. Or maybe he's traumatized nung 2016 elections	hindi siya ang tatay niya bc from what i've learned in my history class, ferdinand marcos is smart unlike bongbong who faked his degree lol. bongbong is also a tax evader. he can't attend on debates and you know why? he has no balls. or maybe he's traumatized nung 2016 elections
@Pontifex @LaityFamilyLife Bongbong Marcos,Sara Zimmerman Duterte and their friends from New York will do anything for Bongbong Marcos and Sara Duterte to win Philippine Presidential and Vice-Presidential elections. And then Bongbong Marcos will become a dictator just like his father-so he can do anything.	bongbong marcos,sara zimmerman duterte and their friends from new york will do anything for bongbong marcos and sara duterte to win philippine presidential and vice-presidential elections. and then bongbong marcos will become a dictator just like his father-so he can do anything.
Reference: Bongbong Marcos website and YouTube. Kuya sir, akala ko po nagresearch kayo. #halalanspace	reference: bongbong marcos website and youtube. kuya sir, akala ko po nagresearch kayo. #halalanspace
tipong matatakot ang mga demons lumapit #PasigLabanParaSaTRoPa #PasigLaban (Photo: Team Leni Robredo) https://t.co/3iwaEeUBtd	tipong matatakot ang mga demons lumapit #pasiglabanparasatropa #pasiglaban (photo: team leni robredo)
norberto gonzales is a whole-ass boomer, and i'm glad that nobody will ever vote for his fascist ass	norberto gonzales is a whole-ass boomer, and i'm glad that nobody will ever vote for his fascist ass
Goodnight! Sarap makipgbardagulan sa mga trolls at minions hahahaha bukas ulit! Pag bastos na mga bibig auto blocked mala-Ping Lacson lang hahahaha. □□	goodnight! sarap makipgbardagulan sa mga trolls at minions hahahaha bukas ulit! pag bastos na mga bibig auto blocked mala-ping lacson lang hahahaha. □□
@lovsyejisu we're facing an election rn and we want Leni Robredo to win (that's why we're using this pink border) against a son of a dictator.	we're facing an election rn and we want leni robredo to win (that's why we're using this pink border) against a son of a dictator.
Hayop na talunang Dilawang fenk Si Fake Vp Leni Robredo, was pa ngang drug test result until now eh https://t.co/ONtXhMpROQ	hayop na talunang dilawang fenk si fake vp leni robredo, was pa ngang drug test result until now eh

C.3. BERT-CNN without Hashtags and with ALL-CAPS

Table 4.3. Test Case for BERT-CNN without Hashtags and with ALL-CAPS

Test Case No. 3
Filename: no_hashtags_with_allcaps - no_hashtags_with_allcaps.pdf

Original Tweet	Cleaned Tweet
KULANG ATA SA TULOG SI NORBERTO GONZALES HAHAHAHAHHAHAHAHAHA SINASABI MO DIYAN	KULANG ATA SA TULOG SI NORBERTO GONZALES HAHAHAHAHHAHAHAHAHA SINASABI MO DIYAN
Hirosii's argument screams misogyny not valid information. The thing here is that you can not defend Bongbong Marcos without blabbering unethical arguments lol.	hirosii's argument screams misogyny not valid information. the thing here is that you can not defend bongbong marcos without blabbering unethical arguments lol.
Who is the better vote? Bongbong #marcos, who is funded by the dictators son, or human rights lawyer Roberdo? The answer is obvious. #TheFilipinoVotes #Voting	who is the better vote? bongbong , who is funded by the dictators son, or human rights lawyer roberdo? the answer is obvious.
@skzoowifey hindi siya ang tatay niya bc from what I've learned in my history class, Ferdinand Marcos is smart unlike Bongbong who faked his degree lol. Bongbong is also a tax evader. He can't attend on debates and you know why? He has no balls. Or maybe he's traumatized nung 2016 elections	hindi siya ang tatay niya bc from what I've learned in my history class, ferdinand marcos is smart unlike bongbong who faked his degree lol. bongbong is also a tax evader. he can't attend on debates and you know why? he has no balls. or maybe he's traumatized nung 2016 elections
@Pontifex @LaityFamilyLife Bongbong Marcos,Sara Zimmerman Duterte and their friends from New York will do anything for Bongbong Marcos and Sara Duterte to win Philippine Presidential and Vice-Presidential elections. And then Bongbong Marcos will become a dictator just like his father-so he can do anything.	bongbong marcos,sara zimmerman duterte and their friends from new york will do anything for bongbong marcos and sara duterte to win philippine presidential and vice-presidential elections. and then bongbong marcos will become a dictator just like his father-so he can do anything.
Reference: Bongbong Marcos website and YouTube. Kuya sir, akala ko po nagresearch kayo. #halalanspace	reference: bongbong marcos website and youtube. kuya sir, akala ko po nagresearch kayo.
tipong matatakot ang mga demons lumapit #PasigLabanParaSaTRoPa #PasigLaban (Photo: Team Leni Robredo) https://t.co/3iwaEeUBtd	tipong matatakot ang mga demons lumapit (photo: team leni robredo)
norberto gonzales is a whole-ass boomer, and i'm glad that nobody will ever vote for his fascist ass	norberto gonzales is a whole-ass boomer, and i'm glad that nobody will ever vote for his fascist ass
Goodnight! Sarap makipgbardagulan sa mga trolls at minions hahahaha bukas ulit! Pag bastos na mga bibig auto blocked mala-Ping Lacson lang hahahaha. □□	goodnight! sarap makipgbardagulan sa mga trolls at minions hahahaha bukas ulit! pag bastos na mga bibig auto blocked mala-ping lacson lang hahahaha. □□
@lovsyejisu we're facing an election rn and we want Leni Robredo to win (that's why we're using this pink border) against a son of a dictator.	we're facing an election rn and we want leni robredo to win (that's why we're using this pink border) against a son of a dictator.
Hayop na talunang Dilawang fenk Si Fake Vp Leni Robredo, was pa ngang drug test result until now eh https://t.co/ONtXhMpROQ	hayop na talunang dilawang fenk si fake vp leni robredo, was pa ngang drug test result until now eh

C.4. BERT-CNN with Hashtags and All-Caps

Table 4.4. Test Case for BERT-CNN with Hashtags and ALL-CAPS

Test Case No. 4
Filename: with_hashtags_allcaps - with_hashtags_allcaps.pdf

Original Tweet	Cleaned Tweet
KULANG ATA SA TULOG SI NORBERTO GONZALES HAHAHAHAHHAHAHAHAHA SINASABI MO DIYAN	KULANG ATA SA TULOG SI NORBERTO GONZALES HAHAHAHAHHAHAHAHAHA SINASABI MO DIYAN
Hirosii's argument screams misogyny not valid information. The thing here is that you can not defend Bongbong Marcos without blabbering unethical arguments lol.	hirosii's argument screams misogyny not valid information. the thing here is that you can not defend bongbong marcos without blabbering unethical arguments lol.
Who is the better vote? Bongbong #marcos, who is funded by the dictators son, or human rights lawyer Roberdo? The answer is obvious. #TheFilipinoVotes #Voting	who is the better vote? bongbong , who is funded by the dictators son, or human rights lawyer roberdo? the answer is obvious.
@skzoowifey hindi siya ang tatay niya bc from what I've learned in my history class, Ferdinand Marcos is smart unlike Bongbong who faked his degree lol. Bongbong is also a tax evader. He can't attend on debates and you know why? He has no balls. Or maybe he's traumatized nung 2016 elections	hindi siya ang tatay niya bc from what I've learned in my history class, ferdinand marcos is smart unlike bongbong who faked his degree lol. bongbong is also a tax evader. he can't attend on debates and you know why? he has no balls. or maybe he's traumatized nung 2016 elections
@Pontifex @LaityFamilyLife Bongbong Marcos,Sara Zimmerman Duterte and their friends from New York will do anything for Bongbong Marcos and Sara Duterte to win Philippine Presidential and Vice-Presidential elections. And then Bongbong Marcos will become a dictator just like his father-so he can do anything.	bongbong marcos,sara zimmerman duterte and their friends from new york will do anything for bongbong marcos and sara duterte to win philippine presidential and vice-presidential elections. and then bongbong marcos will become a dictator just like his father-so he can do anything.
Reference: Bongbong Marcos website and YouTube. Kuya sir, akala ko po nagresearch kayo. #halalanspace	reference: bongbong marcos website and youtube. kuya sir, akala ko po nagresearch kayo.
tipong matatakot ang mga demons lumapit #PasigLabanParaSaTRoPa #PasigLaban (Photo: Team Leni Robredo) https://t.co/3iwaEeUBtd	tipong matatakot ang mga demons lumapit (photo: team leni robredo)
norberto gonzales is a whole-ass boomer, and i'm glad that nobody will ever vote for his fascist ass	norberto gonzales is a whole-ass boomer, and i'm glad that nobody will ever vote for his fascist ass
Goodnight! Sarap makipgbardagulan sa mga trolls at minions hahahaha bukas ulit! Pag bastos na mga bibig auto blocked mala-Ping Lacson lang hahahaha. □□	goodnight! sarap makipgbardagulan sa mga trolls at minions hahahaha bukas ulit! pag bastos na mga bibig auto blocked mala-ping lacson lang hahahaha. □□
@lovsyejisu we're facing an election rn and we want Leni Robredo to win (that's why we're using this pink border) against a son of a dictator.	we're facing an election rn and we want leni robredo to win (that's why we're using this pink border) against a son of a dictator.
Hayop na talunang Dilawang fenk Si Fake Vp Leni Robredo, was pa ngang drug test result until now eh https://t.co/ONtXhMpROQ	hayop na talunang dilawang fenk si fake vp leni robredo, was pa ngang drug test result until now eh

D. System Demonstration of Test Data

This section includes a complete demonstration of the system in detecting hate from the 2022 Philippine presidential election-related posts. The system is derived from a base study by Arganosa et. al. (2022) where they made a system similar to the goal of classifying hate speech in Philippine election-related posts. The key change from this concept being the modification for word embeddings and inclusion of hashtags and ALL-CAPS. The system was implemented using Python as the programming language and Jupyter Notebook for the program structure as well as other imports and libraries mentioned in the previous chapters.

D.1. Main System Process

The demonstration will discuss the process of classifying posts for hate speech with the Binary Classification CNN and Multi Label Classification CNN. The expected output should show the results and performance of the model's classification in posts as either hate or non-hate speech as well as positive, negative or neutral sentiments.

D.1.1. *Data Loading and Extraction*

The data used for this demonstration came from the same dataset used by Arganosa et. al. (2022) containing posts in context of the Philippine elections during the year 2021 to 2022. The dataset was downloaded from the base study team's repository and extracted in Jupyter Notebook from a .csv file.

The Figures 4.7 and 4.8 showcased the different datasets extracted and how they were prepared for preprocessing.

```
data_path = 'binary-dataset.csv'
data_df = pd.read_csv(data_path)
data_df = data_df.rename(columns={'Tweet Content': 'text', 'Sentiment': 'sentiment', 'Label': 'label',})
```

Figure 4.7. Data Loading and Extraction Code Snippet for Binary Classification

```
data_path = 'multilabel-dataset.csv'
data_df = pd.read_csv(data_path)
data_df = data_df.rename(columns={'Tweet Content': 'text', 'Sentiment': 'sentiment', 'Label': 'label'})
```

Figure 4.8. Data Loading and Extraction Code Snippet for Multi Label Classification

The snippet of Figure 4.7 and Figure 4.8 above also showed the extraction method for the data to become readable and accessible to modification in the Jupyter Notebook. The data was segmented into columns namely by text, sentiment, and label.

Table 4.5. Binary Classification Dataset

Binary Classification Dataset

	text	sentiment	label
0	Bongbong Marcos absent pa rin sa Comelec-spons...	Negative	Hate
1	In the Philippines, presidential frontrunner B...	Negative	Hate
2	THE DELUDED ONE\n\nLeni Robredo, a Filipino po...	Negative	Hate
3	@ANCALERTS @_katrinadomingo Walang kaaway at ...	Negative	Hate
4	'The young ones, they don't know': How a dicta...	Negative	Hate
...	...	...	...
5115	LOOK: Dating Sen. Bongbong Marcos, nakikipagpu...	Neutral	Non-hate
5116	Gagi, nahihiwagaan talaga ako sa mga atabs na ...	Neutral	Non-hate
5117	manifesting circle: \n\n 🤞 ...	Neutral	Non-hate
5118	"Ang tatanglaw sa ating bayan ay isang ILAW NG...	Neutral	Non-hate
5119	[2/2] On undocumented OFWs. PRESIDENT LENI ROB...	Neutral	Non-hate

5120 rows × 3 columns

Table 4.6. Multi Label Classification Dataset

Multi Label Classification Dataset

	text	sentiment	label
0	Dictator's son on the cusp of power in the #Ph...	Negative	Hate
1	Kung ang definition ni BongBong Marcos sa isan...	Negative	Hate
2	Leni Robredo is not being subjected by her opp...	Negative	Hate
3	@imstillsour Tuwang tuwa sila sa Pink hahaha.....	Negative	Hate
4	Kapag hindi si VP Leni Robredo ang iboboto mo ...	Negative	Hate
...	...	...	...
7675	VP LENI ROBREDO FOR 2022 https://t.co/2ahSefgmST	Neutral	Non-hate
7676	@alt_ego143 @rapplerdotcom @ramboreports hindi...	Neutral	Non-hate
7677	I can't vote yet, but my president is Leni Rob...	Neutral	Non-hate
7678	Anyways President Leni Robredo #SaveLegendsOfT...	Neutral	Non-hate
7679	Kung ayaw nyo d wag nyo iboto.Napaka simple.. ...	Neutral	Non-hate

7680 rows × 3 columns

In the base study, every post underwent a two-step labeling process. Initially, posts were categorized based on sentiment as either positive, negative, or neutral. Subsequently, each post was further classified as either expressing hate or non-hate, with negative sentiments being associated with hate and positive/neutral sentiments falling under the non-hate category. In the case of the multi label dataset, there were 2,560 posts for each sentiment category. Meanwhile, the binary dataset also comprised 2,560 posts, where non-hate posts encompassed a combination of 1,280 positive sentiments and 1,280 neutral sentiments. The grouping by label was to ensure that the model is fed with equal proportions of information to conclude a concrete learning for each type of classification. Table 4.6. above visualized the sample extracted data in a dataframe format for both binary and multi label datasets.

D.1.2 Preprocessing

Once the data was extracted, it was ready to be preprocessed and trained into the model. By following the steps from the Preprocessing module, the Figure 4.1.10 below demonstrates how the data is cleaned and filtered for any unnecessary and misleading values.

D.1.2.1 De Identification and URL Removal

```
# Date De-Identification and URL Removal
def remove_mentions_and_url(text):
    mention_pattern = re.compile(r'@\w+')
    url_pattern = re.compile(r'https?://\S+|www\.\S+')

    # Use re.sub to remove mentions
    cleaned_text = mention_pattern.sub('', text)

    # Use re.sub to replace URLs with an empty string
    cleaned_text = url_pattern.sub('', cleaned_text)

    # Remove extra spaces and strip leading/trailing spaces
    cleaned_text = ' '.join(cleaned_text.split())

    return cleaned_text
```

Figure 4.9. Mention and URL Removal Function

The *remove_mentions_and_url()* function served to identify and remove any format of account mentions such as *@sovereign1249* or *@unityEnjoyer* within the X post as well as URLs which lead users to other sites or posts. The researchers speculated that both account mentions and URLs hold no significance in context for the model to learn and classify with presidential-election related posts.

D.1.2.2 Special Character Processing

```python
# Special Character Processing
def remove_special_characters(text):
    text = emoji.replace_emoji(text, replace="[emoji]")

    # Split the text into words
    words = text.split(" ")

    # Initialize an empty string to store the cleaned text
    cleaned_text = ""

    # Iterate through each word
    for word in words:
        # Check if the word contains only special characters or "[emoji]"
        if not (re.match(r"^[_\W]+$", word) or "[emoji]" in word):
            if len(cleaned_text) == 0:
                cleaned_text = f"{word}"
            else:
                cleaned_text = f"{cleaned_text} {word}"

    # Remove diacritics
    text_no_diacritics = unidecode.unidecode(cleaned_text)

    # Split the text into words
    sentence = text_no_diacritics.split(" ")
    output = ""

    # Remove special characters and numerics
    for part in sentence:
        part = re.sub("[^A-Za-z ]+$", "", part)
        part = re.sub("^[^A-Za-z #]+", "", part)
        if not (len(part) <= 1 or re.match(r"[^a-zA-Z#]", part)):
            if len(output) == 0:
                output = f"{part}"
            else:
                output = f"{output} {part}"

    # Remove extra spaces and strip leading/trailing spaces
    cleaned_text = ' '.join(output.split())

    return cleaned_text
```

Figure 4.10. Special Character Processing

This *remove_special_characters()* function performed a series of operations to remove special characters. It first replaced emojis in the text with a placeholder, "[emoji]". Then, it splits the text into individual words and iterates through each word, removing those that consist solely of special characters or the "[emoji]" placeholder. The function further removed diacritics (accented characters) from the cleaned text. It then went through another process to eliminate special characters and numerics from the

beginning and end of each word. This function did not remove hashtag symbols, providing researchers with the flexibility to decide whether to retain or remove hashtags based on the requirements of the chosen model. Table 4.8 below shows the output of the preprocessed dataset as compared to the original dataset.

Table 4.7. Preprocessed Dataset

Original Dataset
Text: kakampinks sleeping soundly tonight knowing that leni robredo did well in tonight’s debate. pero yung iba, double time s a paghahanap ng mali. 😂 hirap talaga pag wala ka mapagmalaki sa kandidato mo sa iba ka nalang titingin. #10RobredoPresident #I panaloNa10To Sentiment: Negative --- Text: Fascinating read 👇 Why Bongbong Marcos, a Philippine dictator’s Son, leads the race for the presidency https://t.co/U93PcvdMYB Sentiment: Negative
Preprocessed Dataset
Text: kakampinks sleeping soundly tonight knowing that leni robredo did well in tonight's debate pero yung iba double time sa paghahanap ng mali hirap talaga pag wala ka mapagmalaki sa kandidato mo sa iba ka nalang titingin #10RobredoPresident #IpanaloN a10To Sentiment: Negative --- Text: Fascinating read Why Bongbong Marcos Philippine dictator's Son leads the race for the presidency Sentiment: Negative

D.1.2.3. Hashtags Processing

Once the researchers produced the desired output after preprocessing the raw dataset, the necessary features were then implemented based on the test cases for both binary and multi label data. Starting with hashtags, the researchers have the option to use this function for BERT-CNN and BERT-CNN with ALL-CAPS models.

```
# Hashtag Removal
def remove_hashtags(text):
    # Split the text into words
    words = text.split()

    # Initialize an empty list to store cleaned words
    cleaned_words = []

    for word in words:
        # Check if the word is a hashtag (starts with #)
        if not word.startswith('#'):
            cleaned_words.append(word)

    # Join the cleaned words into a single string
    cleaned_text = ' '.join(cleaned_words)

    return cleaned_text
```

Figure 4.11. Hashtags Processing

D.1.2.4. Normalization (with ALL-CAPS Processing)

The normalization process involved converting the text into lowercase format. Under this process, there's also ALL-CAPS retainment, wherein every text was converted except for ALL-CAPS. Like hashtag processing, the researchers have the option to choose lowercase or lowercase except ALL-CAPS.

```
# Lowercase
data_df['text'] = data_df['text'].str.lower()
```

Figure 4.12. Normalization (Lowercase)

```python
# Lowercase except ALL-CAPS
def lowercase_except_all_caps(text):
    cleaned_text = remove_special_characters(text)

    words = cleaned_text.split()
    filtered_words = []

    for word in words:
        if word.isupper() and not word.istitle():
            filtered_words.append(word)
        else:
            filtered_words.append(re.sub(r'([A-Z][a-z]+)', lambda x: x.group(1).lower(), word))

    return ' '.join(filtered_words)
```

Figure 4.13. ALL-CAPS Processing (Lowercase except ALL-CAPS)

This preprocessing step was implemented to improve the model's runtime and overall performance with uniformity as well the option to preserve ALL-CAPS. The table 4.9 below shows the binary and multi label datasets with the necessary features for each test case namely, *BERT-CNN without Hashtags and ALL-CAPS, BERT-CNN with Hashtags, BERT-CNN with ALL-CAPS, and BERT-CNN with Hashtags and ALL-CAPS.*

Table 4.8. Preprocessed Binary and Multi Label Dataset with Test Cases

Binary BERT-CNN without Hashtags and ALL-CAPS
Text: kahit anong gawin mong pagpapalit ng kulay leni robredo,hindi mo na maalis ma-erase sa inyong katauhan ang pagiging dila wan,tatak dilawan ka talaga,yun yong nakatatak na rin sa aming isipan bilang mga mamayan at mga voters Label: Hate ~~~~~~ Text: pano ba yan eh mismong si ping lacson isang red tagger Label: Hate ~~~~~~ Text: magiging author na si bongbong marcos magsusulat na sya ng saln accounting equation nya debit minus credit equals kupit anong accounting book kaya ito Label: Hate ~~~~~~
Multi Label BERT-CNN without Hashtags and ALL-CAPS

```
Text:  kung ang definition ni bongbong marcos sa isang terorista ay isang armadong grupo na nananakit nanunupil ng civilian tee
nager na non combatant pinapatunayan lang na ang pnp-afp ay terorista ejk sa drug war at martial law mismo ang ebidensya

Sentiment:  Negative

--------------------------------------------------

Text:  leni robredo is not being subjected by her opponents to gender attact but her intelligence and capacity to lead are put
into question we do not want puppet president

Sentiment:  Negative

--------------------------------------------------

Text:  tuwang tuwa sila sa pink hahaha..hindi nila alam pink color is sign for bongbong marcos..kasi favorite nya isuot pink...
ambilis ni leni mag grab ng color kasi nakita nila langit nag pink

Sentiment:  Negative
```

Binary BERT-CNN with Hashtags

```
Text:  norberto gonzales or bleng blong it's tie for me

Label:  Hate

--------------------------------------------------

Text:  ulol yung mga nag sasabing i'm an inc and vote for leni robredo ulol sino niloko nyo kapwa nyo kakampink hahaha pag lili
naw lang po sa lahat na yung mga nag popost ng ganyan is hindi talaga inc thank you and godbless #bbmsara #inc

Label:  Hate

--------------------------------------------------

Text:  walang sense ng reality si ping the blocker lacson

Label:  Hate
```

Multi Label BERT-CNN with Hashtags

```
Text:  kapag hindi si vp leni robredo ang iboboto mo sa may years kang mamalasin

Sentiment:  Negative

--------------------------------------------------

Text:  kakampinks sleeping soundly tonight knowing that leni robredo did well in tonight's debate pero yung iba double time sa
paghahanap ng mali hirap talaga pag wala ka mapagmalaki sa kandidato mo sa iba ka nalang titingin #10robredopresident #ipanalon
a10to

Sentiment:  Negative

--------------------------------------------------

Text:  leni robredo daw potanginaaaaausasassasa

Sentiment:  Negative
```

Binary BERT-CNN with ALL-CAPS

```
Text:  marcos jr received INC's bloc votes as VP in but lost to leni robredo's smartmagic shenanigans there corrected it for th
em

Label:  Hate

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Text:  kapag hindi si VP leni robredo ang iboboto mo sa may YEARS KANG MAMALASIN

Label:  Hate

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Text:  dahil maraming naniniwala at kasama na mga bata ni boy alamano na hindi corrupt si VP LENI ROBREDO lenikikoalltheway

Label:  Hate

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Multi Label BERT-CNN with ALL-CAPS

```
Text:  kung ang definition ni bongbong marcos sa isang terorista ay isang armadong grupo na nananakit nanunupil ng civilian tee
nager na NON COMBATANT pinapatunayan lang na ang PNP-AFP ay terorista EJK sa drug war at martial law mismo ang ebidensya

Sentiment:  Negative

-------------------------------------------------

Text:  leni robredo is not being subjected by her opponents to gender attact but her intelligence and capacity to lead are put
into question we do not want puppet president

Sentiment:  Negative

-------------------------------------------------

Text:  tuwang tuwa sila sa pink hahaha..hindi nila alam pink color is sign for bongbong marcos..kasi favorite nya isuot pink...
ambilis ni leni mag grab ng color kasi nakita nila langit nag pink

Sentiment:  Negative
```

Binary BERT-CNN with Hashtags and ALL-CAPS

```
Text:  Ang toxic nanaman ng social media battle of PINKLAWANS AND UNITEAM SUPPORTERS Sana pag katapos ng election kung sino man
g manalong presidente support nalang naten administration nila Bongbong ako pero kung manalo si leni Kiko Manny or ping lacson
#Election2022PH

Label:  Hate

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Text:  People say ang hirap intindihin ni Leni Robredo pag nagsasalita Minsan naisip ko dapat she uses conversational statement
s Pero naisip ko din na ang hindi makaintindi sa kanya ang tunay na bobo at sayang ang boto #LeniRobredoForPresident #LeniKikoT
eam

Label:  Hate

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Text:  Salute to you Leni Robredo You will be our next President in section Lugaw classroom

Label:  Hate

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Multi Label BERT-CNN with Hashtags and ALL-CAPS

```
Text:  kung ang definition ni bongbong marcos sa isang terorista ay isang armadong grupo na nananakit nanunupil ng civilian tee
nager na NON COMBATANT pinapatunayan lang na ang PNP-AFP ay terorista EJK sa drug war at martial law mismo ang ebidensya #mangg
agawaVSmagnanakaw

Sentiment:  Negative

--------------------------------------------------

Text:  leni robredo is not being subjected by her opponents to gender attact but her intelligence and capacity to lead are put
into question we do not want puppet president

Sentiment:  Negative

--------------------------------------------------

Text:  tuwang tuwa sila sa pink hahaha..hindi nila alam pink color is sign for bongbong marcos..kasi favorite nya isuot pink...
ambilis ni leni mag grab ng color kasi nakita nila langit nag pink

Sentiment:  Negative
```

After the dataset's text was thoroughly cleaned and processed, it was then tokenized in preparation for the next stage of the system.

D.1.2.5. Tokenization

The *tokenize_text()* function that leveraged a BERT tokenizer to process a given text. Tokenization involved breaking down the input text into smaller units called tokens. The BERT tokenizer was applied to the input text. The resulting tokens were then truncated to accommodate the maximum length allowed by the BERT model, accounting for the special [CLS] (classification) and [SEP] (separator) tokens. Finally, the function converted the tokens into their corresponding token IDs using the tokenizer's *convert_tokens_to_ids* method. This process was crucial for preparing text data to be compatible with BERT-based models, which operate on tokenized representations of text rather than raw text.

```python
# Tokenization
def tokenize_text(text):
    tokens = tokenizer.tokenize(text)
    tokens = tokens[:tokenizer.model_max_length - 2]  # Account for [CLS] and [SEP] tokens
    indexed_tokens = tokenizer.convert_tokens_to_ids(tokens)
    return indexed_tokens
```

Figure 4.14. Tokenization

Table 4.9. Tokenized Dataset

Tokenized Data Output
Text: [2022, 8583, 6583, 4241, 2863, 2319, 7842, 4259, 2457, 21368, 2080, 10556, 21368, 16098, 14753, 18259, 5063, 14810, 4487, 5003, 2912, 4502, 21693, 21818, 2006, 16405, 3217, 10556, 8275, 2739] Label: Hate ~~~~~~ Text: [4496, 21201, 24334, 18798, 2072, 6487, 23417, 17712, 2080, 3892, 2089, 2417, 1011, 5210, 9033, 16768, 2721, 7842, 1041, 15992, 3277, 9033, 14397, 2386, 2004, 2467, 1037, 2100, 18996, 11905, 19313, 9033, 2003, 3683, 1037, 2100, 3364, 21368, 16098, 9033, 18749, 3385, 6583, 23877, 9152, 3148, 17712, 2080, 7842, 2529, 2916, 3980, 9033, 10125, 27871, 2953, 6583, 23877, 9152, 3148, 17712, 2080, 7842, 2217, 9006, 8163, 9152, 3148] Label: Hate ~~~~~~ Text: [21687, 2290, 9033, 24829, 5283, 3148, 11085, 6790, 6583, 4487, 26234, 22854, 2290, 16660, 6292, 8486, 10556, 5332, 24547, 2050, 4487, 3406, 6415, 2050, 15544, 3406, 10556, 4487, 9587, 26234, 9033, 21210, 18798, 2072, 6487, 23417, 17076, 14477, 3070, 10722, 12274, 10052, 16634, 2290, 10556, 3736, 5620, 16098, 2078, 6583, 24547, 5654, 16405, 2863, 9739, 11493, 7842, 4937, 5685, 13860, 2229, 2012, 6583, 2290, 9874] Label: Hate

The Table 4.9 above showed the final output for the preprocessing module. Once the preprocessed data was tokenized, the researchers moved on to the splitting module stage of the system.

D.1.3 Splitting

The data was then split as seen on the code snippets of Figure 4.15, Figure 4.16, and Figure 4.17. below. The researchers decided to utilize the splitting in either 70:30, 80:20 or 90:10 portion in order to encapsulate the BERT-CNN model's understanding and recognition between *Hate* and *Non-Hate* as well as *Positive, Negative* and *Neutral*.

```
train_df, test_df = train_test_split(data_df, test_size=0.3, random_state=SEED)
```

Figure 4.15. Code Snippet of 70:30 Data Splitting

```
train_df, test_df = train_test_split(data_df, test_size=0.2, random_state=SEED)
```

Figure 4.16. Code Snippet of 80:20 Data Splitting

```
train_df, test_df = train_test_split(data_df, test_size=0.1, random_state=SEED)
```

Figure 4.17. Code Snippet of 90:10 Data Splitting

D.1.4. BERT Word Embeddings

The next module required the team to initialize the BERT embedding model. The BERT word embeddings module was instrumental in capturing rich contextual representations of words. During preprocessing, the input text was tokenized, and this module took the tokenized data as input, providing word embeddings that encapsulate contextual information learned during pre-training. The extracted BERT token embeddings are then seamlessly integrated into the embedding layer of the Binary and Multi Label CNN models.

```
with torch.no_grad():
    for i, token in enumerate(tokenizer.get_vocab()):
        token_id = tokenizer.convert_tokens_to_ids(token)
        token_embedding = bert_model.embeddings.word_embeddings.weight[token_id]
        model.embedding.weight[i].data.copy_(token_embedding)
```

Figure 4.18. BERT Word Embeddings

D.1.5 Binary and Multi Label CNN

The BERT token embeddings were then fed as input for the CNN models. The researches initialized the model through an exploration of different applications of

Convolutional Neural Networks for test cases or extraction of features as well as detecting patterns with hate or non-hate speech.

```python
class CNN(nn.Module):
    def __init__(self, vocab_size, embedding_dim, n_filters, filter_sizes, output_dim, dropout):
        super().__init__()

        self.embedding = nn.Embedding(vocab_size, embedding_dim)
        self.embedding_dropout = nn.Dropout(dropout)

        # Convolutional layers with varying kernel sizes
        self.convs = nn.ModuleList([
            nn.Conv1d(in_channels=embedding_dim, out_channels=n_filters, kernel_size=fs, padding='same')
            for fs in filter_sizes
        ])

        # Batch normalization for each convolutional layer
        self.bns = nn.ModuleList([nn.BatchNorm1d(n_filters) for _ in filter_sizes])

        # Global Max Pooling
        self.global_pooling = nn.AdaptiveMaxPool1d(1)

        # Dropout
        self.dropout = nn.Dropout(dropout)

        # Fully connected layer
        self.fc = nn.Linear(len(filter_sizes) * n_filters, output_dim)

    def forward(self, x):
        embedded = self.embedding(x)
        embedded = self.embedding_dropout(embedded)
        x = embedded.permute(0, 2, 1)

        # Apply convolutional and batch normalization layers
        x = [F.relu(bn(conv(x))) for conv, bn in zip(self.convs, self.bns)]
        x = [self.global_pooling(conv).squeeze(2) for conv in x]

        # Concatenate the feature maps
        x = torch.cat(x, 1)

        x = self.dropout(x)
        x = self.fc(x)

        return x
```

Figure 4.19. CNN Model for Binary

```python
class CNN(nn.Module):
    def __init__(self, vocab_size, embedding_dim, n_filters, filter_sizes, num_classes, dropout):
        super().__init__()

        self.embedding = nn.Embedding(vocab_size, embedding_dim)
        self.embedding_dropout = nn.Dropout(dropout)

        # Convolutional layers with varying kernel sizes
        self.convs = nn.ModuleList([
            nn.Conv1d(in_channels=embedding_dim, out_channels=n_filters, kernel_size=fs, padding='same')
            for fs in filter_sizes
        ])

        # Batch normalization for each convolutional layer
        self.bns = nn.ModuleList([nn.BatchNorm1d(n_filters) for _ in filter_sizes])

        # Global Max Pooling
        self.global_pooling = nn.AdaptiveMaxPool1d(1)

        # Dropout
        self.dropout = nn.Dropout(dropout)

        # Fully connected layer
        self.fc = nn.Linear(len(filter_sizes) * n_filters, num_classes)

    def forward(self, x):
        embedded = self.embedding(x)
        embedded = self.embedding_dropout(embedded)
        x = embedded.permute(0, 2, 1)

        # Apply convolutional and batch normalization layers
        x = [F.relu(bn(conv(x))) for conv, bn in zip(self.convs, self.bns)]
        x = [self.global_pooling(conv).squeeze(2) for conv in x]

        # Concatenate the feature maps
        x = torch.cat(x, 1)

        x = self.dropout(x)
        x = self.fc(x)

        return x
```

Figure 4.20. CNN Model for Multi Label

The CNN model architecture was adaptable to both binary and multi label classification tasks. It was much more complex than the base study's CNN model. It was initiated with an embedding layer, followed by an embedding dropout layer. The heart of the model comprised convolutional layers with varying kernel sizes. Each convolutional

layer was paired with batch normalization, fostering stable and effective training. The global max pooling was then employed to distill the most salient features from the convolutional layers, creating a fixed-size representation of the input sequence. Subsequently, a dropout layer contributed to regularization, preventing the co-adaptation of feature detectors. The final fully connected layer processed the concatenated feature maps, mapping them to the specified output dimension.

For binary classification, the fully connected layer had an output dimension of 1, allowing the model to produce a single output. A sigmoid activation function was applied to the output, giving a probability score between 0 and 1. In the case of multi label classification, the fully connected layer was configured with an output dimension corresponding to the number of classes which is three (3), employing a softmax activation function to generate class probabilities across all classes.

During the forward pass, the input underwent embedding, dropout, and permutation operations, followed by the application of convolutional and batch normalization layers. The global max pooling and concatenation steps prepared the features for the fully connected layer, which produced the final output of the model.

D.1.6. Evaluation

After initializing the BERT-CNN model for both Binary and Multi-Label systems, the model was then ready to be evaluated by the necessary metrics and confusion matrix to measure the model's performance in detecting *Hate* and *Non-Hate* speech, as well as sentiment classes such as *Positive, Negative, and Neutral* in the case of sentiment classification.

```
# Print other metrics (e.g., accuracy, loss) for the current epoch
print(f'Epoch: {epoch + 1:02}\n')
print(f'\tTrain Loss: {train_loss:.3f}')
print(f'\tTest Loss: {test_loss:.3f}\n')
print(f'\tAccuracy: {accuracy:.4f} | F1-Score: {f1:.4f}')
print(f'\tPrecision: {precision:.4f} | Recall: {recall:.4f}\n')
print(f'\tClassification Report:\n')
report = classification_report(true_sentiments, predicted_sentiments,
                               target_names=['Positive', 'Negative', 'Neutral'], digits = 4)
print(f'{report}\n')
```

Figure 4.21. Model Evaluation Metrics

The Figure 4.1.21 above showed how the model's performance was measured through *accuracy*, *precision*, *recall*, and *F1-Score* and assisted to determine the model's proficiency in detecting *Hate Speech* in comparison to the base study's *fastText* CNN model. The classification reports of the system were presented like in Figure 4.1.22 below.

```
Classification Report:

              precision    recall  f1-score   support

           0     0.8899    0.8592    0.8743       781
           1     0.8593    0.8901    0.8744       755

    accuracy                         0.8743      1536
   macro avg     0.8746    0.8746    0.8743      1536
weighted avg     0.8749    0.8743    0.8743      1536
```

Figure 4.22. Classification Report Sample

E. Results and Analysis

The detailed outcomes derived from the experimental evaluation of the BERT-CNN model in hate speech detection within Philippine election-related posts is observed. The *fastText* CNN models were also employed by the researchers to ensure an accurate comparison between this study's model and the base study. The results are presented across different models and their corresponding splits, providing a comprehensive analysis of the model's performance in various configurations.

E.1. *fastText* CNN Binary Classification

The researchers implemented *fastText* CNN with the base study's preprocessing techniques and dataset. However, in this study's model implementation, the dataset was shuffled differently due to the specific manner in which it was split. The different training and testing split rations are 70:30, 80:20, and 90:10. This section focuses on evaluating the performance of the *fastText* CNN model in the context of binary classification, distinguishing between *Hate* and *Non-Hate* categories under these various split configurations.

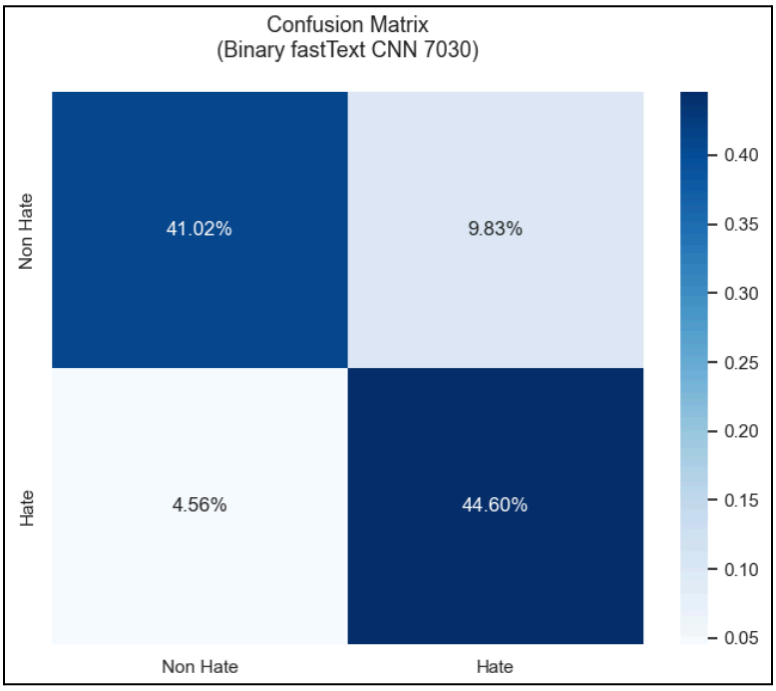
E.1.1. 70:30 Split

```
---------- Evaluation on Test Data ----------
              precision    recall  f1-score   support

           0     0.9000    0.8067    0.8508       781
           1     0.8194    0.9073    0.8611       755

    accuracy                         0.8561      1536
   macro avg     0.8597    0.8570    0.8559      1536
weighted avg     0.8604    0.8561    0.8558      1536
```

Figure 4.23. Classification Report (fastText CNN 70:30)



630	151
70	685

Figure 4.24. Confusion Matrix (fastText CNN 70:30)

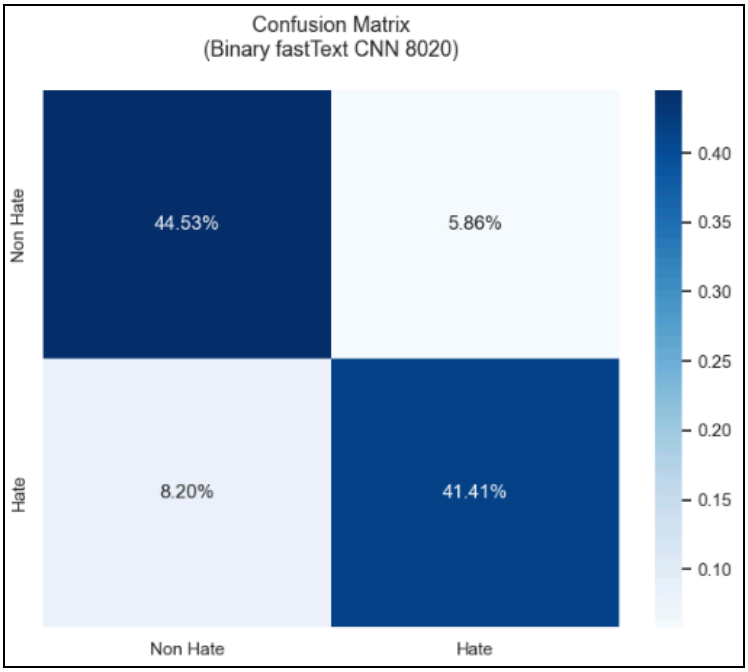
The *fastText* CNN model for binary label classification achieved its highest results across all splitting ratio in detecting hate speech at an accuracy of 85.61%, precision of 85.97%, recall of 85.70%, and an F1-Score of 85.59%. Additionally, the model had exhibited the highest metric performance in terms of precision at a 70:30 split ratio.

E.1.2. 80:20 Split

---------- Evaluation on Test Data ----------

	precision	recall	f1-score	support
0	0.8444	0.8837	0.8636	516
1	0.8760	0.8346	0.8548	508
accuracy			0.8594	1024
macro avg	0.8602	0.8592	0.8592	1024
weighted avg	0.8601	0.8594	0.8593	1024

Figure 4.25. Classification Report (fastText CNN 80:20)



456	60
84	424

Figure 4.26. Confusion Matrix (fastText CNN 80:20)

This split for the model achieved its best results in detecting hate speech at an accuracy of 85.94%, precision of 86.02%, recall of 85.92%, and an F1-Score of 85.92%. Compared to the 70:30 split ratio of the model, this split exhibited the highest metric performance in terms of precision at a 80:20 split ratio.

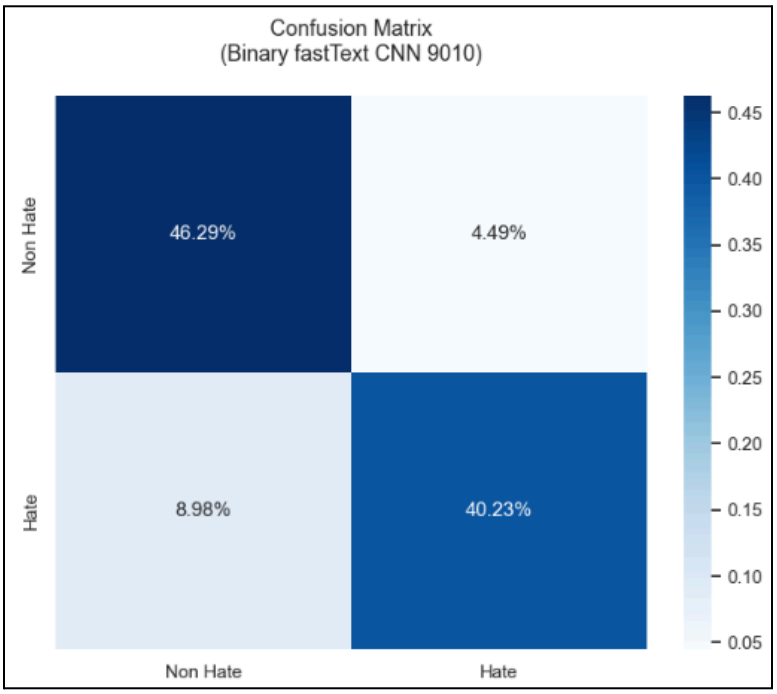
E.1.3. 90:10 Split

```
---------- Evaluation on Test Data ----------
              precision    recall  f1-score   support

           0     0.8375    0.9115    0.8729       260
           1     0.8996    0.8175    0.8565       252

    accuracy                         0.8652       512
   macro avg     0.8685    0.8645    0.8647       512
weighted avg     0.8680    0.8652    0.8649       512
```

Figure 4.27. Classification Report (fastText CNN 90:10)



237	23
46	206

Figure 4.28. Confusion Matrix (fastText CNN 90:10)

This split for the model achieved its best results in detecting hate speech at an accuracy of 86.52%, precision of 86.85%, recall of 86.45%, and an F1-Score of 86.47%. Moreover, results have shown that this split also exhibited the highest metric performance in terms of precision at a 80:20 split ratio.

E.2. *fastText* CNN Multi Label Classification

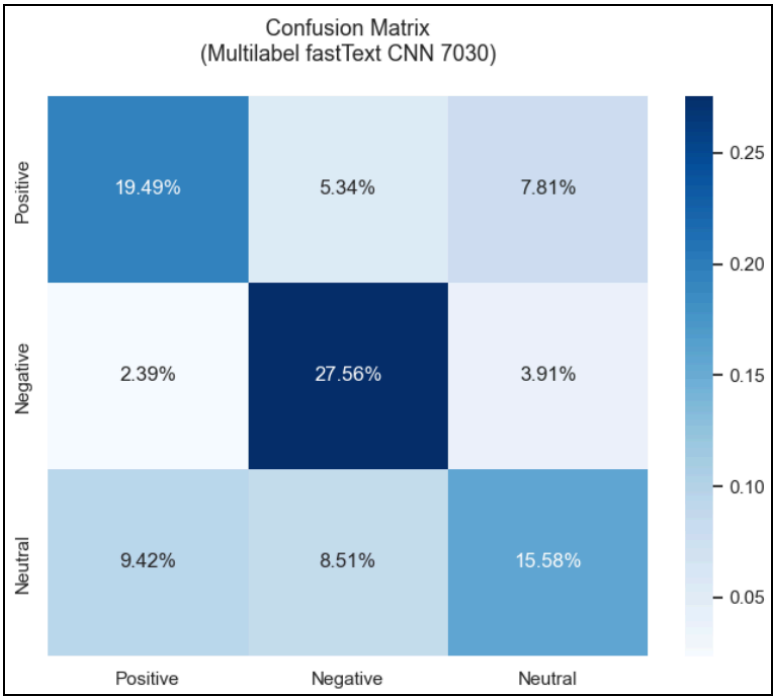
The researchers incorporated the *fastText* CNN model, utilizing the preprocessing techniques and dataset from the base study. This section was dedicated to assessing the performance of the *fastText* CNN model with regards to multi label classification, specifically categorizing sentiments into *Positive, Negative,* and *Neutral* classes. The evaluation was conducted across different split configurations to provide a comprehensive understanding of the model's performance.

E.2.1. 70:30 Split

---------- Evaluation on Test Data ----------

	precision	recall	f1-score	support
0	0.6227	0.5971	0.6096	752
1	0.6656	0.8141	0.7324	780
2	0.5707	0.4650	0.5125	772
accuracy			0.6263	2304
macro avg	0.6197	0.6254	0.6182	2304
weighted avg	0.6198	0.6263	0.6187	2304

Figure 4.29. Classification Report (fastText CNN 70:30)



449	123	180
55	635	90
217	196	359

Figure 4.30. Confusion Matrix (fastText CNN 70:30)

The 70:30 split ratio for the *fastText* CNN model for multi label classification achieved its best performance in detecting hate speech at an accuracy of 62.63%, precision of 61.97%, recall of 62.54%, and an F1-Score of 61.82%. Results have shown that this split also exhibited the highest metric performance in terms of accuracy.

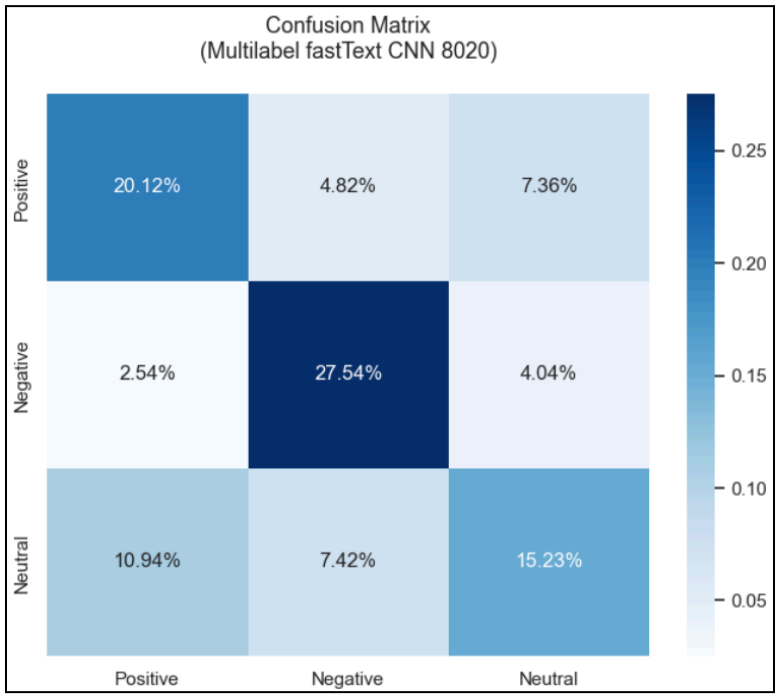
E.2.2. 80:20 Split

```
---------- Evaluation on Test Data ----------
              precision    recall  f1-score   support

           0     0.5988    0.6230    0.6107       496
           1     0.6923    0.8073    0.7454       524
           2     0.5721    0.4535    0.5059       516

    accuracy                         0.6289      1536
   macro avg     0.6211    0.6279    0.6207      1536
weighted avg     0.6218    0.6289    0.6214      1536
```

Figure 4.31. Classification Report (fastText CNN 80:20)



309	74	113
39	423	62
168	114	234

Figure 4.32. Confusion Matrix (fastText CNN 80:20)

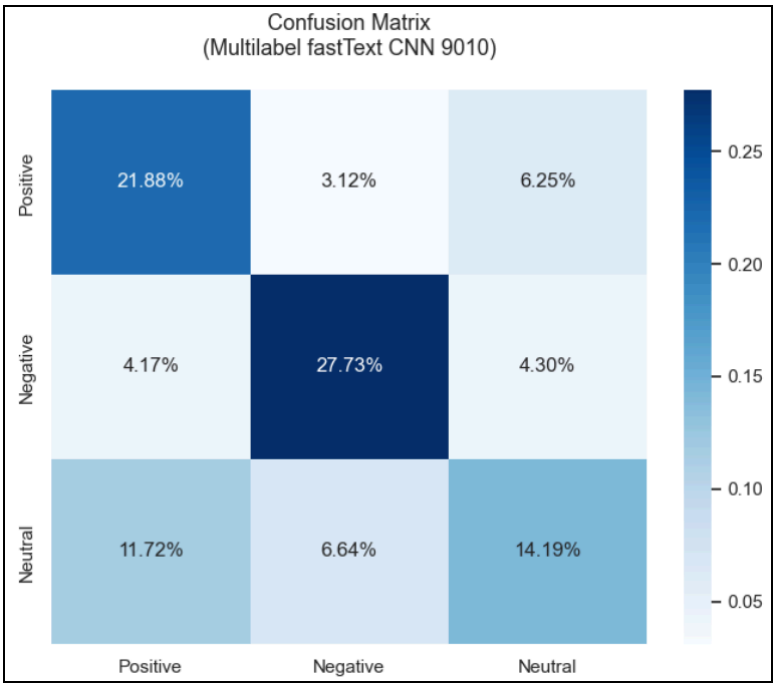
This split ratio for the model for multi label classification achieved its best performance in detecting hate speech at an accuracy of 62.89%, precision of 62.11%, recall of 62.79%, and an F1-Score of 62.07%. The results for this model have shown that its highest metric performance is also focused in terms of accuracy.

E.2.3. 90:10 Split

---------- Evaluation on Test Data ----------

	precision	recall	f1-score	support
0	0.5793	0.7000	0.6340	240
1	0.7396	0.7662	0.7527	278
2	0.5737	0.4360	0.4955	250
accuracy			0.6380	768
macro avg	0.6309	0.6341	0.6274	768
weighted avg	0.6355	0.6380	0.6318	768

Figure 4.33. Classification Report (fastText CNN 90:10)



168	24	48
32	213	33
90	51	109

Figure 4.34. Confusion Matrix (fastText CNN 90:10)

The *fastText* CNN model for multi label classification achieved its best performance in detecting hate speech at an accuracy of 63.80%, precision of 63.09%, recall of 63.41%, and an F1-Score of 62.74%. This shows that the model performs best in terms of precision across all split ratios.

E.3. BERT-CNN Binary Classification

The performance of the BERT-CNN model in discerning *Hate* speech from *Non-Hate* speech was thoroughly examined. This scrutiny encompasses an exploration of the model's results under various configurations, including the integration of hashtags, ALL-CAPS, a combination of hashtags and ALL-CAPS, and enhanced preprocessing techniques. Additionally, the investigation delves into the implications of different

training and testing split ratios (70:30, 80:20, and 90:10), providing valuable insights into the optimal conditions for training and evaluating the model's proficiency in binary classification.

E.3.1. BERT-CNN

The BERT-CNN model, serving as the base system model for this study, employed uncased BERT, signifying that the model did not differentiate between uppercase and lowercase letters during processing. The decision to utilize the uncased version of BERT aligned with the preprocessing strategy of text normalization, specifically converting all text to lowercase. This preprocessing step was employed to ensure consistency and reduce the dimensionality of the vocabulary, as it treated words in uppercase and lowercase forms as equivalent. This model did not include hashtags or ALL-CAPS.

This model will be compared to the *fastText* CNN model by Arganosa et. al. (2022) in terms of the model's accuracy, precision, recall and F1-Score. The best epoch as well as the highest metrics result will be mentioned per training and testing split ratio.

E.3.1.1. 70:30 Split

```
Epoch: 03

        Train Loss: 0.195
        Test Loss: 0.306
```

Figure 4.35. Train-Test Loss (BERT-CNN 70:30)

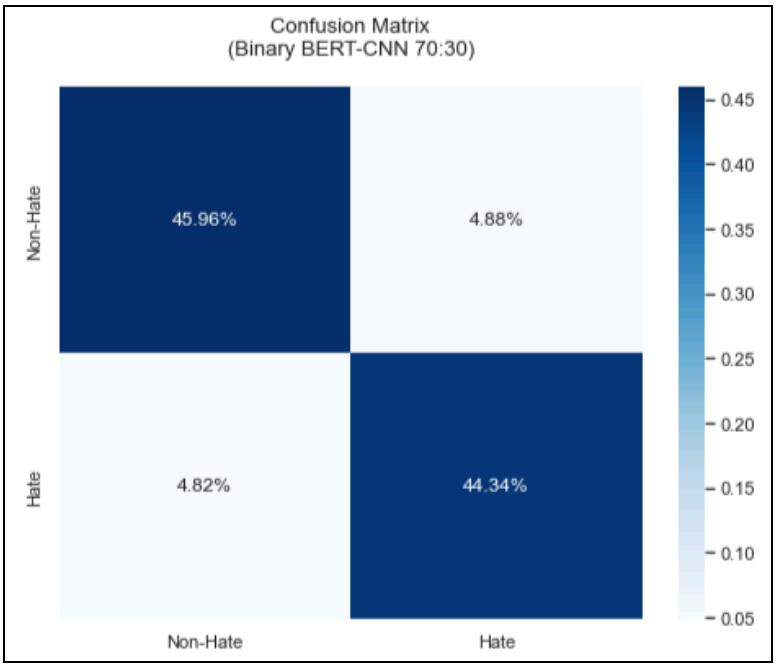
```
Classification Report:

              precision    recall  f1-score   support

           0     0.9051    0.9040    0.9045       781
           1     0.9008    0.9020    0.9014       755

    accuracy                         0.9030      1536
   macro avg     0.9030    0.9030    0.9030      1536
weighted avg     0.9030    0.9030    0.9030      1536
```

Figure 4.36. Classification Report (BERT-CNN 70:30)



706	75
74	681

Figure 4.37. Confusion Matrix (BERT-CNN 70:30)

The BERT-CNN model without the inclusion of hashtags (HT) and ALL-CAPS (AC) achieved its best results in the third epoch. At its lowest test loss, the model demonstrated an accuracy of 90.30%, precision of 90.30%, recall of 90.30%, and an F1-Score of 90.30%. Notably, the model exhibited the highest performance in terms of accuracy at a 70:30 split ratio.

E.3.1.2. 80:20 Split

```
Epoch: 03

        Train Loss: 0.196
        Test Loss: 0.291
```

Figure 4.38. Train-Test Loss (BERT-CNN 80:20)

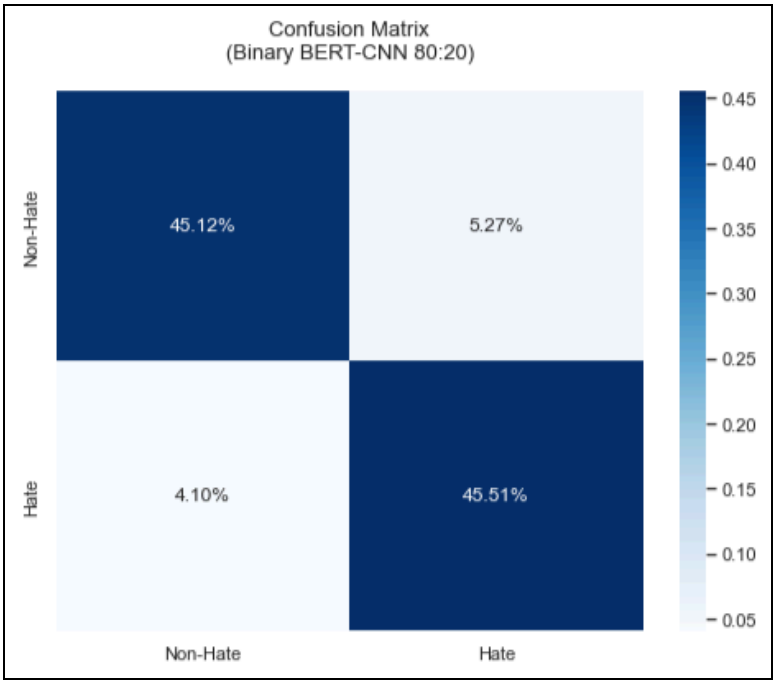
```
Classification Report:

              precision    recall  f1-score   support

           0     0.9167    0.8953    0.9059       516
           1     0.8962    0.9173    0.9066       508

    accuracy                         0.9062      1024
   macro avg     0.9064    0.9063    0.9062      1024
weighted avg     0.9065    0.9062    0.9062      1024
```

Figure 4.39. Classification Report (BERT-CNN 80:20)



462	54
42	466

Figure 4.40. Confusion Matrix (BERT-CNN 80:20)

The BERT-CNN model without HT + AC at 80:20 split ratio also achieved the best results at its third span or epoch. At its lowest test loss, the model produced an output with 90.62% on accuracy, 90.64% on precision, 90.63% on recall and 90.62% on F1-Score. Out of all the metrics, the model achieved the highest with recall at 80:20 split ratio.

E.3.1.3. 90:10 Split

```
Epoch: 03

        Train Loss: 0.184
        Test Loss: 0.234
```

Figure 4.41. Train-Test Loss (BERT-CNN 90:10)

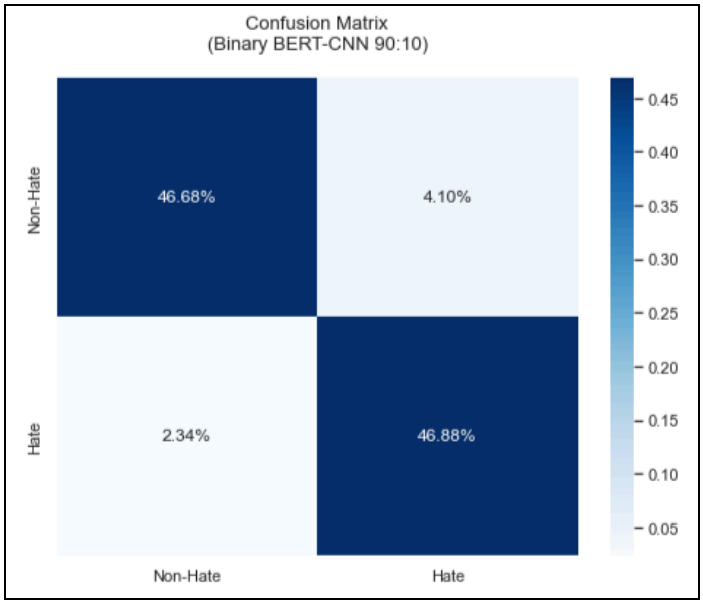
```
Classification Report:

              precision    recall  f1-score   support

           0     0.9522    0.9192    0.9354       260
           1     0.9195    0.9524    0.9357       252

    accuracy                         0.9355       512
   macro avg     0.9359    0.9358    0.9355       512
weighted avg     0.9361    0.9355    0.9355       512
```

Figure 4.42. Classification Report (BERT-CNN 90:10)



239	21
12	240

Figure 4.43. Confusion Matrix (BERT-CNN 90:10)

The BERT-CNN model without HT and AC achieved the best results at its third epoch. At its lowest test loss, the model produced an output with 93.55% on accuracy, 91.95% on precision, 95.24% on recall and 93.57% on F1-Score. Out of all the metrics, the model also achieved the highest with recall at 90:10 split ratio.

E.3.2. BERT-CNN with Hashtags

Upon recommendation of the authors of the base study, the researchers incorporated the use of hashtags in posts as an approach to improve context and sentiment analysis using BERT-CNN algorithms. Given the case insensitivity of hashtags on X, the decision was made to utilize BERT uncased. This section discussed the performance of BERT-CNN with the inclusion of only hashtags (HT) as well as its best epoch and evaluation metric across all split ratios.

E.3.2.1. 70:30 Split

```
Epoch: 03

        Train Loss: 0.199
        Test Loss: 0.306
```

Figure 4.44. Train-Test Loss (BERT-CNN w/ HT 70:30)

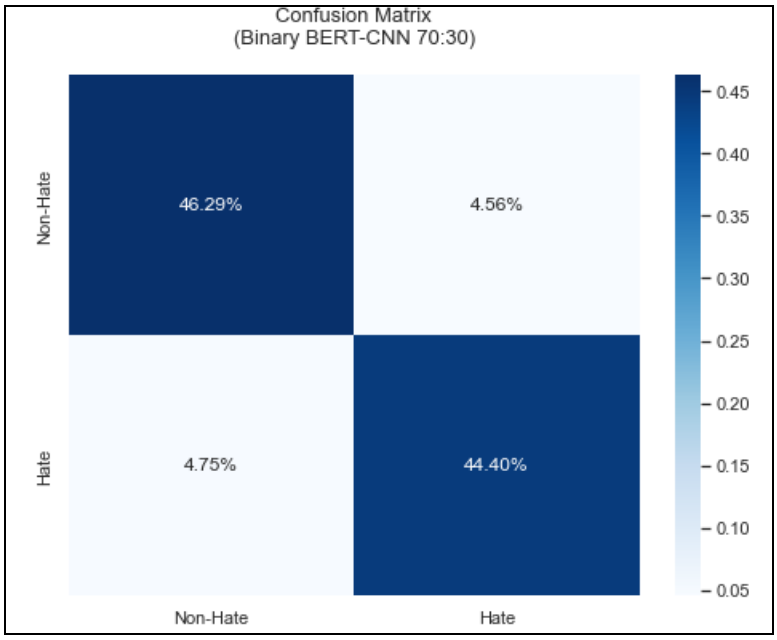
```
Classification Report:

              precision    recall  f1-score   support

           0     0.9069    0.9104    0.9086       781
           1     0.9069    0.9033    0.9051       755

    accuracy                         0.9069      1536
   macro avg     0.9069    0.9068    0.9069      1536
weighted avg     0.9069    0.9069    0.9069      1536
```

Figure 4.45. Classification Report (BERT-CNN w/ HT 70:30)



711	70
73	682

Figure 4.46. Classification Report (BERT-CNN w/ HT 70:30)

The BERT-CNN model with HT achieved the best results at epoch 3. At its lowest test loss, the model produced an output with 90.69% accuracy, 90.69% precision, 90.68% recall and 90.69% F1-Score. Out of all the metrics, the model also achieved the highest with recall at 70:30 split ratio.

E.3.2.2. 80:20 Split

```
Epoch: 03

        Train Loss: 0.204
        Test Loss: 0.274
```

Figure 4.47. Train-Test Loss (BERT-CNN w/ HT 80:20)

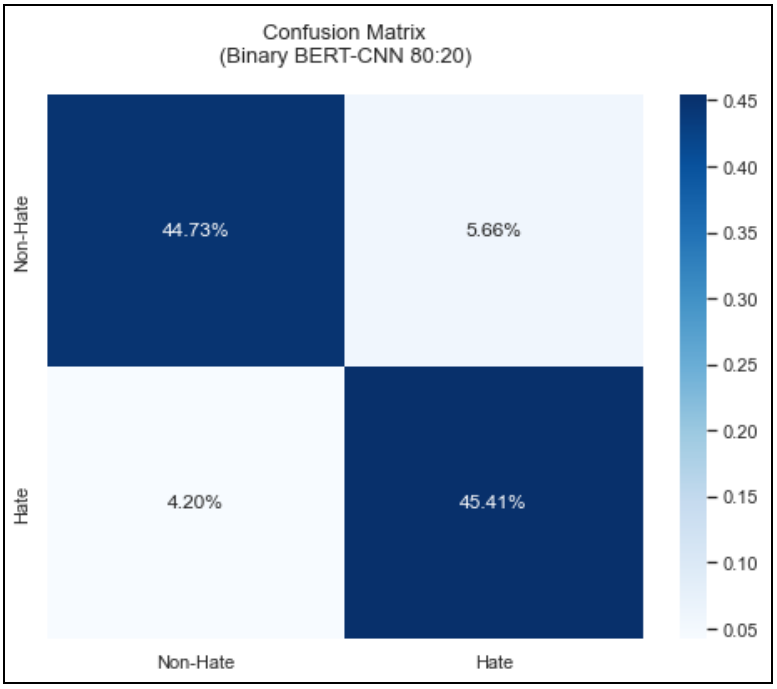
```
Classification Report:

              precision    recall  f1-score   support

           0     0.9142    0.8876    0.9007       516
           1     0.8891    0.9154    0.9020       508

    accuracy                         0.9014      1024
   macro avg     0.9016    0.9015    0.9014      1024
weighted avg     0.9017    0.9014    0.9014      1024
```

Figure 4.48. Classification Report (BERT-CNN w/ HT 80:20)



458	58
43	465

Figure 4.49. Confusion Matrix (BERT-CNN w/ HT 80:20)

The BERT-CNN model with HT achieved the best results at epoch 3 as well. At its lowest test loss, the model produced an output with 88.96% accuracy, 88.99% precision, 88.98% recall and 88.96% F1-Score. Out of all the metrics, the model also achieved the highest with recall at 80:20 split ratio.

E.3.2.3. 90:10 Split

```
Epoch: 03

        Train Loss: 0.184
        Test Loss: 0.239
```

Figure 4.50. Train-Test Loss (BERT-CNN w/ HT 90:10)

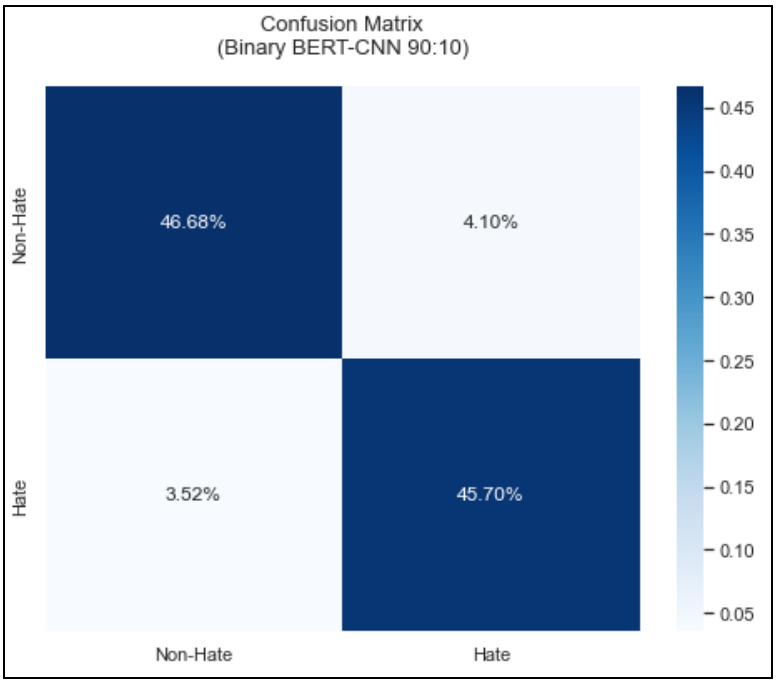
```
Classification Report:

              precision    recall  f1-score   support

           0     0.9300    0.9192    0.9246       260
           1     0.9176    0.9286    0.9231       252

    accuracy                         0.9238       512
   macro avg     0.9238    0.9239    0.9238       512
weighted avg     0.9239    0.9238    0.9238       512
```

Figure 4.51. Classification Report (BERT-CNN w/ HT 90:10)



239	21
18	234

Figure 4.52. Confusion Matrix (BERT-CNN w/ HT 90:10)

The final split for the BERT-CNN model with HT achieved the best results at epoch 3. At its lowest test loss, the model produced an output with 92.38% accuracy, 92.38% precision, 92.39% recall and 92.38% F1-Score. Out of all the metrics, the model still achieved the highest with recall at 90:10 split ratio.

E.3.3. BERT-CNN with ALL-CAPS

The researchers also considered the potential impact of words or phrases in ALL-CAPS, recognizing them as potential conversational cues with underlying meaning in sentiment analysis. Given the importance of retaining the case information for ALL-CAPS, BERT cased was employed, allowing the model to distinguish between uppercase and lowercase letters. This section presents the results of the BERT-CNN

model with the inclusion of ALL-CAPS in posts. The model is analyzed across different split ratios, highlighting the best epoch and the highest evaluation metric in classification performance.

E.3.3.1. 70:30 Split

```
Epoch: 03

        Train Loss: 0.284
        Test Loss: 0.308
```

Figure 4.53. Train-Test Loss (BERT-CNN w/ AC 70:30)

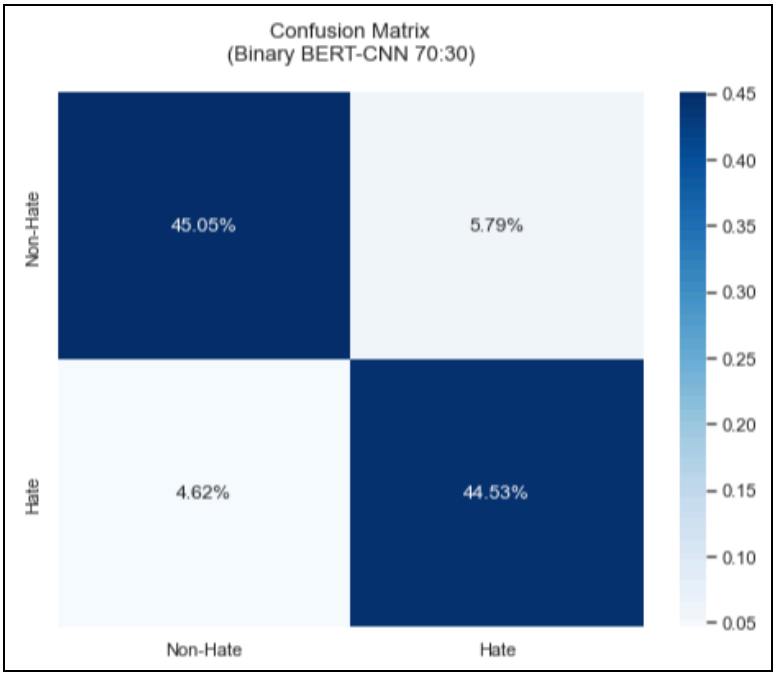
```
Classification Report:

              precision    recall  f1-score   support

           0     0.9069    0.8860    0.8964       781
           1     0.8849    0.9060    0.8953       755

    accuracy                         0.8958      1536
   macro avg     0.8959    0.8960    0.8958      1536
weighted avg     0.8961    0.8958    0.8958      1536
```

Figure 4.54. Classification Report (BERT-CNN w/ AC 70:30)



692	89
71	684

Figure 4.55. Confusion Matrix (BERT-CNN w/ AC 70:30)

This split for the BERT-CNN model with the inclusion of AC achieved the best results at epoch 3. At its lowest test loss, the model produced an output with 89.58% accuracy, 88.59% precision, 89.60% recall and 89.58% F1-Score. Out of all the metrics, the model achieved the highest with recall at 70:30 training and testing split ratio.

E.3.3.2. 80:20 Split

```
Epoch: 03

        Train Loss: 0.244
        Test Loss: 0.315
```

Figure 4.56. Train-Test Loss (BERT-CNN w/ AC 80:20)

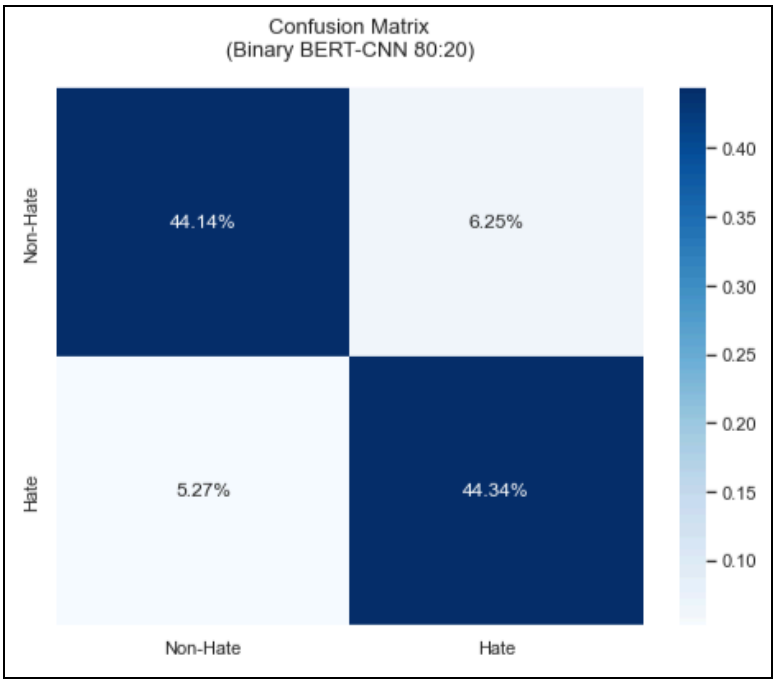
```
Classification Report:

              precision    recall  f1-score   support

           0     0.8933    0.8760    0.8845       516
           1     0.8764    0.8937    0.8850       508

    accuracy                         0.8848      1024
   macro avg     0.8849    0.8848    0.8848      1024
weighted avg     0.8849    0.8848    0.8848      1024
```

Figure 4.57. Classification Report (BERT-CNN w/ AC 80:20)



452	64
54	454

Figure 4.58. Confusion Matrix (BERT-CNN w/ AC 80:20)

This split for the model achieved the best results at epoch 3 as well. At its lowest test loss, the model produced an output with 88.48% accuracy, 88.49% precision, 88.48% recall and 88.48% F1-Score.

E.3.3.3. 90:10 Split

```
Epoch: 03

        Train Loss: 0.291
        Test Loss: 0.281
```

Figure 4.59. Train-Test Loss (BERT-CNN w/ AC 90:10)

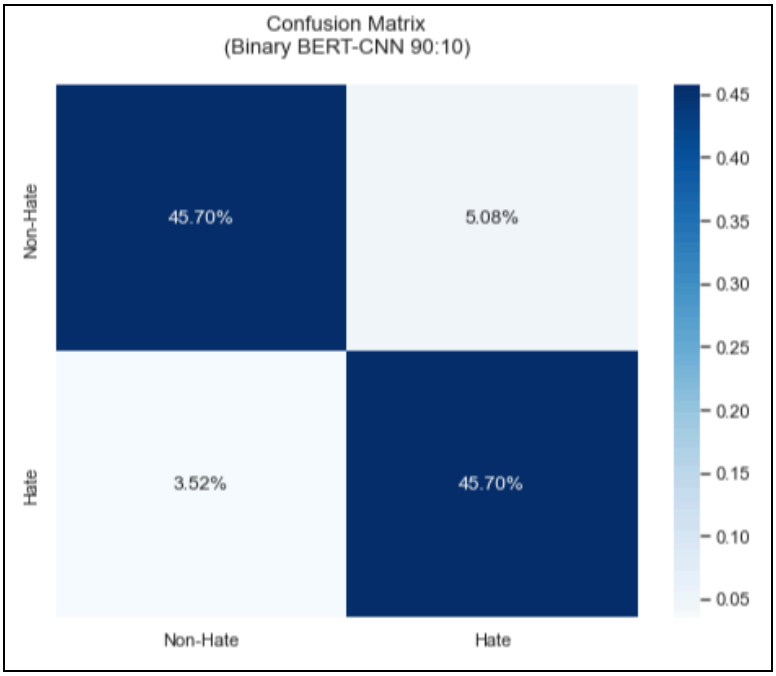
```
Classification Report:

              precision    recall  f1-score   support

           0     0.9286    0.9000    0.9141       260
           1     0.9000    0.9286    0.9141       252

    accuracy                         0.9141       512
   macro avg     0.9143    0.9143    0.9141       512
weighted avg     0.9145    0.9141    0.9141       512
```

Figure 4.60. Classification Report (BERT-CNN w/ AC 90:10)



234	26
18	234

Figure 4.61. Confusion Matrix (BERT-CNN w/ AC 90:10)

The final split for the model achieved the best results at epoch 3. At its lowest test loss, the model produced an output with 91.41% accuracy, 90.00% precision, 92.86% recall and 91.41% F1-Score. Out of all the metrics, the model achieved the highest with recall as well at 90:10 split ratio.

E.3.4. BERT-CNN with Hashtags and ALL-CAPS

The final iteration of the system includes both HT and AC into the BERT-CNN model which was compared with the BERT-CNN model without both features. BERT cased was used to retain case distinctions, especially for ALL-CAPS. The section will elaborate on the best epoch, the highest evaluation metric, and their respective values in the performance report for the BERT-CNN model with HT and AC.

E.3.4.1. 70:30 Split

```
Epoch: 03

        Train Loss: 0.300
        Test Loss: 0.283
```

Figure 4.62. Train-Test Loss (BERT-CNN w/ HT+AC 70:30)

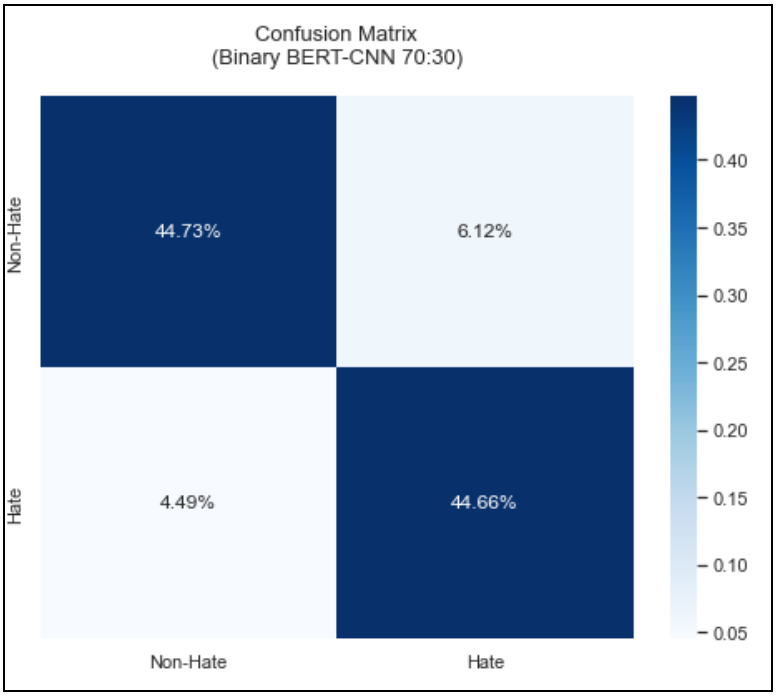
```
Classification Report:

              precision    recall  f1-score   support

           0     0.9087    0.8796    0.8939       781
           1     0.8795    0.9086    0.8938       755

    accuracy                         0.8939      1536
   macro avg     0.8941    0.8941    0.8939      1536
weighted avg     0.8944    0.8939    0.8939      1536
```

Figure 4.63. Classification Report (BERT-CNN w/ HT+AC 70:30)



687	94
69	686

Figure 4.64. Confusion Matrix (BERT-CNN w/ HT+AC 70:30)

The test and training split for this model achieved the best classification results at epoch 3. At its lowest test loss, the model produced an output with 89.39% accuracy, 89.41% precision, 89.41% recall and 89.39% F1-Score. Out of all the metrics, the model achieved the highest with precision and recall at 70:30 training and testing split ratio.

E.3.4.2. 80:20 Split

```
Epoch: 04

        Train Loss: 0.175
        Test Loss: 0.269
```

Figure 4.65. Performance Measures (BERT-CNN w/ HT+AC 80:20)

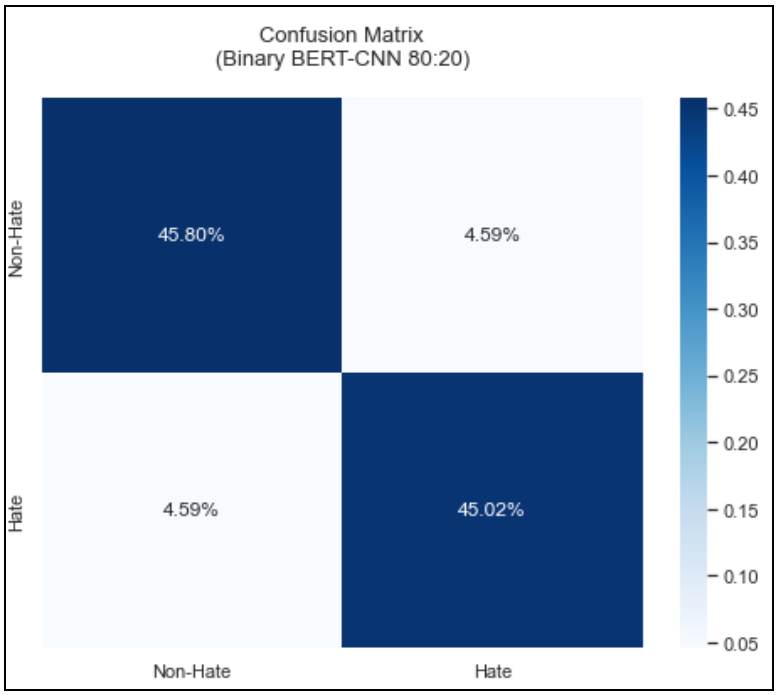
```
Classification Report:

              precision    recall  f1-score   support

           0     0.9089    0.9089    0.9089       516
           1     0.9075    0.9075    0.9075       508

    accuracy                         0.9082      1024
   macro avg     0.9082    0.9082    0.9082      1024
weighted avg     0.9082    0.9082    0.9082      1024
```

Figure 4.66. Classification Report (BERT-CNN w/ HT+AC 80:20)



469	47
47	461

Figure 4.67. Confusion Matrix (BERT-CNN w/ HT+AC 80:20)

The split for this model achieved the best results at epoch 4 compared to the 70:30 split's best performance at epoch 3. At its lowest test loss, the model produced an output with 90.82% accuracy, 90.82% precision, 90.82% recall and 90.82% F1-Score. For all of the metrics, the model achieved the same at 80:20 training and testing split ratio.

E.3.4.3. 90:10 Split

```
Epoch: 03

        Train Loss: 0.278
        Test Loss: 0.250
```

Figure 4.68. Performance Measures (BERT-CNN w/ HT+AC 90:10)

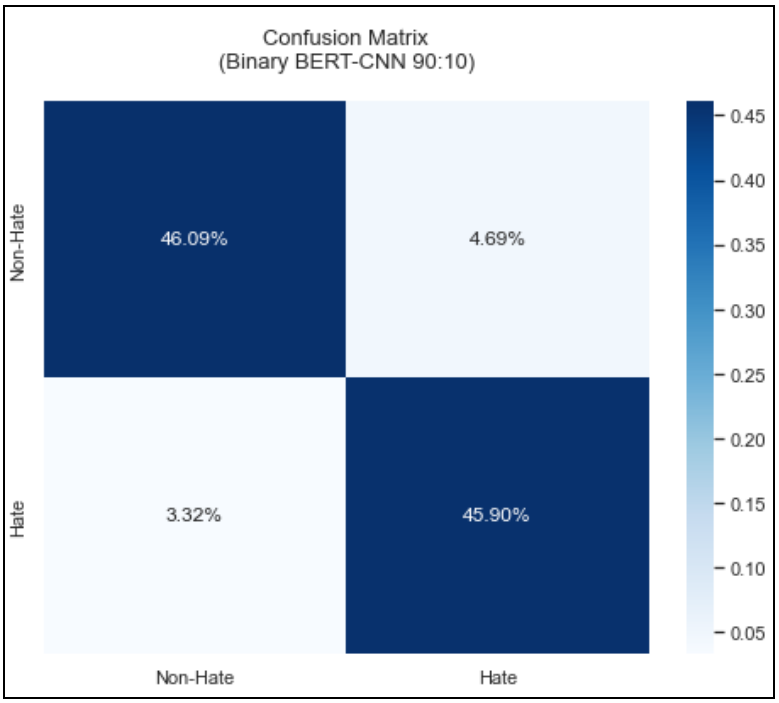
```
Classification Report:

              precision    recall  f1-score   support

           0     0.9328    0.9077    0.9201       260
           1     0.9073    0.9325    0.9198       252

    accuracy                         0.9199       512
   macro avg     0.9201    0.9201    0.9199       512
weighted avg     0.9203    0.9199    0.9199       512
```

Figure 4.69. Classification Report (BERT-CNN w/ HT+AC 90:10)



236	24
17	235

Figure 4.70. Confusion Matrix (BERT-CNN w/ HT+AC 90:10)

The final split for this model achieved the best results at epoch 3. At its lowest test loss, the model produced an output with 91.99% accuracy, 92.01% precision, 92.01% recall and 91.99% F1-Score. For most of the metrics, the model achieved the same highest with precision recall at 90:10 training and testing split ratio.

E.4. BERT-CNN Multi Label Classification

The BERT-CNN model's ability to assign sentiment labels namely, *Positive, Negative, or Neutral,* was assessed. This comprehensive assessment involved scrutinizing the model's performance under various configurations, encompassing the incorporation of Hashtags, ALL-CAPS, and a combination of Hashtags and ALL-CAPS. The objective was to gauge the model's effectiveness in capturing nuanced sentiments within Philippine

election-related posts. The analysis went deeper into exploring how different training and testing split ratios (70:30, 80:20, and 90:10) impact the model, providing valuable insights into the optimal conditions for training and evaluating the BERT-CNN model's multi label classification capabilities.

E.4.1. BERT-CNN

The base system for the BERT-CNN model did not include the use of HT and AC. For the multi label approach, the model will classify text from the three labels namely, *Positive, Negative*, or *Neutral*. Like the Binary BERT-CNN, BERT uncased was utilized, treating the text in a case-insensitive manner. This section discussed the results and best output of each model and their highest result at each split ratio.

E.4.1.1. 70:30 Split

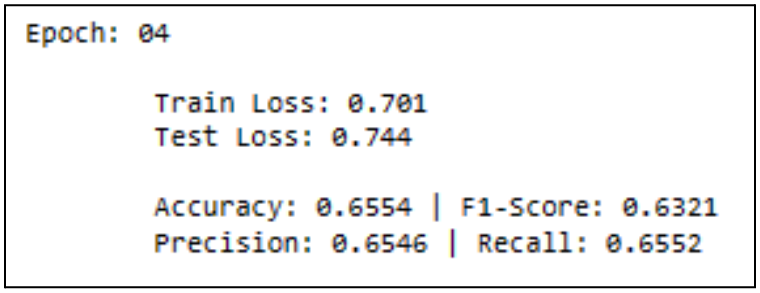
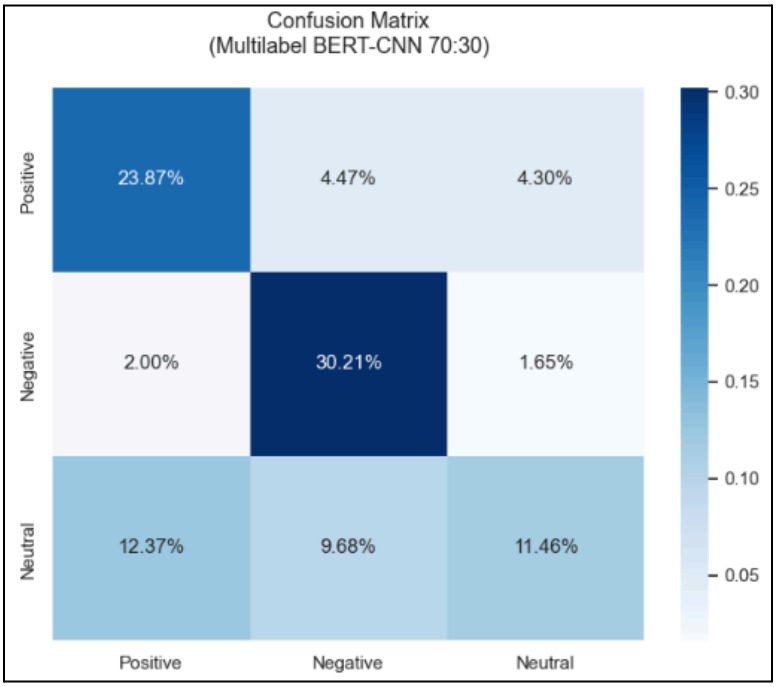


Figure 4.71. Performance Measures (BERT-CNN 70:30)

Classification Report:

	precision	recall	f1-score	support
Positive	0.6243	0.7314	0.6736	752
Negative	0.6810	0.8923	0.7725	780
Neutral	0.6584	0.3420	0.4501	772
accuracy			0.6554	2304
macro avg	0.6546	0.6552	0.6321	2304
weighted avg	0.6549	0.6554	0.6322	2304

Figure 4.72. Classification Report (BERT-CNN 70:30)



550	103	99
46	696	38
285	223	264

Figure 4.73. Confusion Matrix (BERT-CNN 70:30)

The split ratio for this model achieved the best results at the fourth epoch. At its lowest test loss, the model produced an output with 65.54% accuracy, 65.46% precision, 65.52% recall and 63.21% F1-Score. For most of the metrics, the model achieved the same highest with accuracy at 70:30 training and testing split ratio.

E.4.1.2. 80:20 Split

```
Epoch: 04

        Train Loss: 0.651
        Test Loss: 0.751

        Accuracy: 0.6628 | F1-Score: 0.6449
        Precision: 0.6582 | Recall: 0.6624
```

Figure 4.74. Performance Measures (BERT-CNN 80:20)

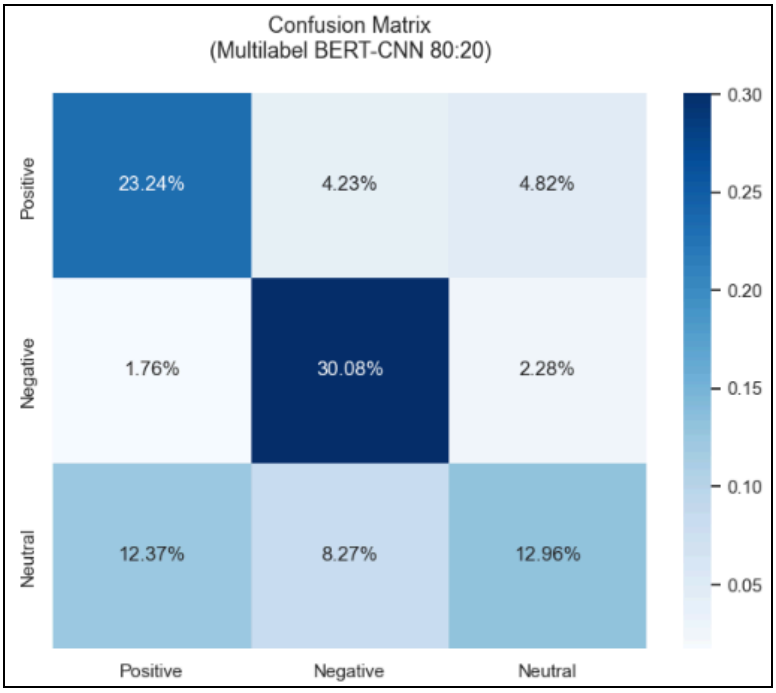
```
Classification Report:

              precision    recall  f1-score   support

    Positive     0.6220    0.7198    0.6673       496
    Negative     0.7064    0.8817    0.7844       524
     Neutral     0.6461    0.3857    0.4830       516

    accuracy                         0.6628      1536
   macro avg     0.6582    0.6624    0.6449      1536
weighted avg     0.6589    0.6628    0.6453      1536
```

Figure 4.75. Classification Report (BERT-CNN 80:20)



357	65	74
27	462	35
190	127	199

Figure 4.76. Confusion Matrix (BERT-CNN 80:20)

The split ratio for this model also achieved the best results at epoch 4. At its lowest test loss, the model produced an output with 66.28% accuracy, 65.82% precision, 66.24% recall and 64.49% F1-Score. For most of the metrics, the model achieved the close highest with accuracy at 80:20 split ratio.

E.4.1.2. 90:10 Split

```
Epoch: 04

        Train Loss: 0.625
        Test Loss: 0.700

        Accuracy: 0.6784 | F1-Score: 0.6627
        Precision: 0.6660 | Recall: 0.6710
```

Figure 4.77. Performance Measures (BERT-CNN 90:10)

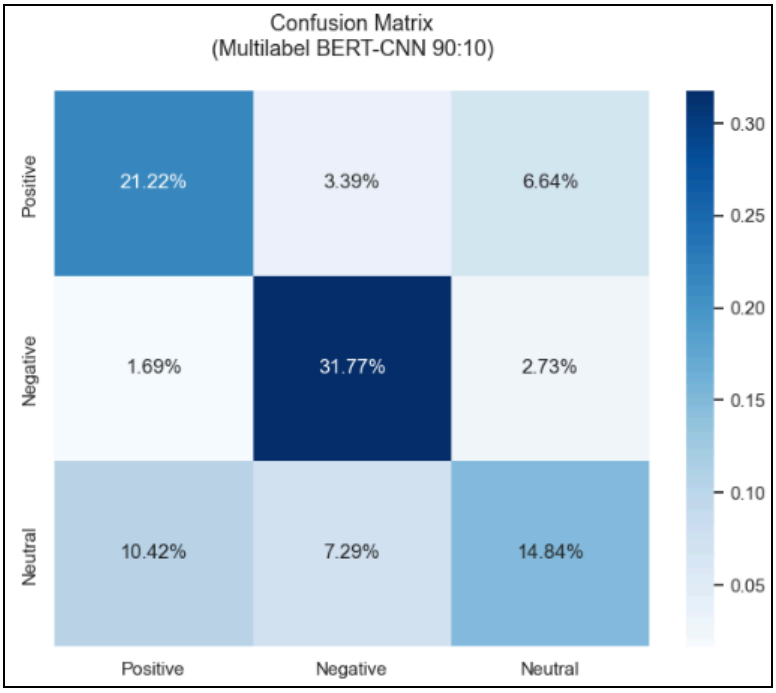
```
Classification Report:

              precision    recall  f1-score   support

    Positive     0.6367    0.6792    0.6573       240
    Negative     0.7485    0.8777    0.8079       278
     Neutral     0.6129    0.4560    0.5229       250

    accuracy                         0.6784       768
   macro avg     0.6660    0.6710    0.6627       768
weighted avg     0.6694    0.6784    0.6681       768
```

Figure 4.78. Classification Report (BERT-CNN 90:10)



163	26	51
13	244	21
80	56	114

Figure 4.79. Confusion Matrix (BERT-CNN 90:10)

The split ratio for this model achieved the best results at epoch 4. At its lowest test loss, the model produced an output with 67.84% accuracy, 66.60% precision, 67.10% recall and 66.27% F1-Score. For the metrics, the model achieved the highest result with accuracy at 90:10 split ratio.

E.4.2. BERT-CNN with Hashtags

The BERT-CNN model with the multi label approach also included the use of HT only, leveraging the uncased version of BERT. This inclusion was theorized to enhance the model's overall performance. This subsection showed the results of the model's classification and its highest evaluation metric with the feature of hashtags.

E.4.2.1. 70:30 Split

```
Epoch: 05

        Train Loss: 0.567
        Test Loss: 0.713

        Accuracy: 0.6793 | F1-Score: 0.6690
        Precision: 0.6737 | Recall: 0.6790
```

Figure 4.80. Performance Measures (BERT-CNN w/ HT 70:30)

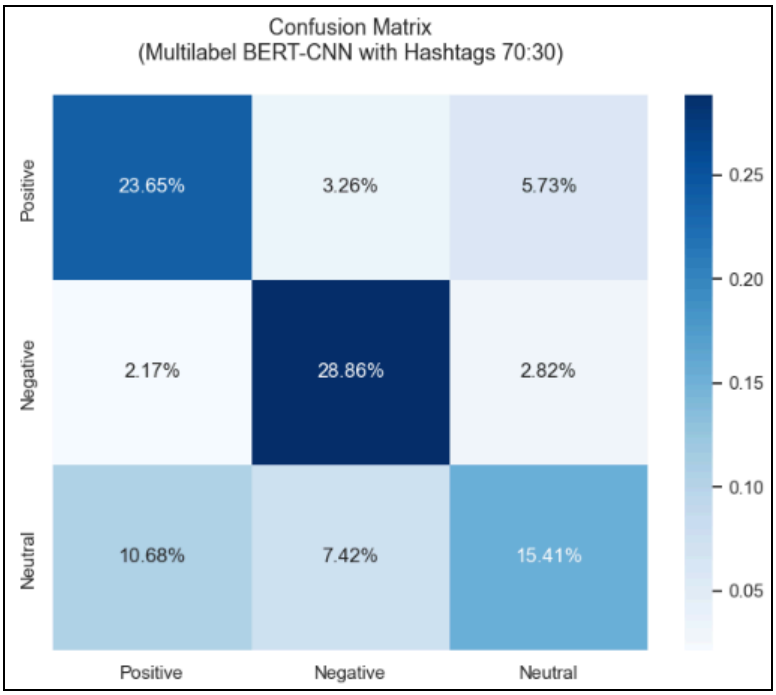
```
Classification Report:

              precision    recall  f1-score   support

    Positive     0.6480    0.7247    0.6842       752
    Negative     0.7300    0.8526    0.7865       780
     Neutral     0.6431    0.4598    0.5363       772

    accuracy                         0.6793      2304
   macro avg     0.6737    0.6790    0.6690      2304
weighted avg     0.6741    0.6793    0.6693      2304
```

Figure 4.81. Classification Report (BERT-CNN w/ HT 70:30)



545	75	132
50	665	65
246	171	355

Figure 4.82. Confusion Matrix (BERT-CNN w/ HT 70:30)

The split ratio for this model achieved the best results at epoch 5. At its lowest test loss, the model produced an output with 67.93% accuracy, 67.37% precision, 67.90% recall and 66.90% F1-Score. Across all of the metrics, the model achieved the highest result with accuracy at 70:30 split ratio.

E.2.2.2. 80:20 Split

```
Epoch: 04

        Train Loss: 0.666
        Test Loss: 0.727

        Accuracy: 0.6706 | F1-Score: 0.6530
        Precision: 0.6712 | Recall: 0.6709
```

Figure 4.83. Performance Measures (BERT-CNN w/ HT 80:20)

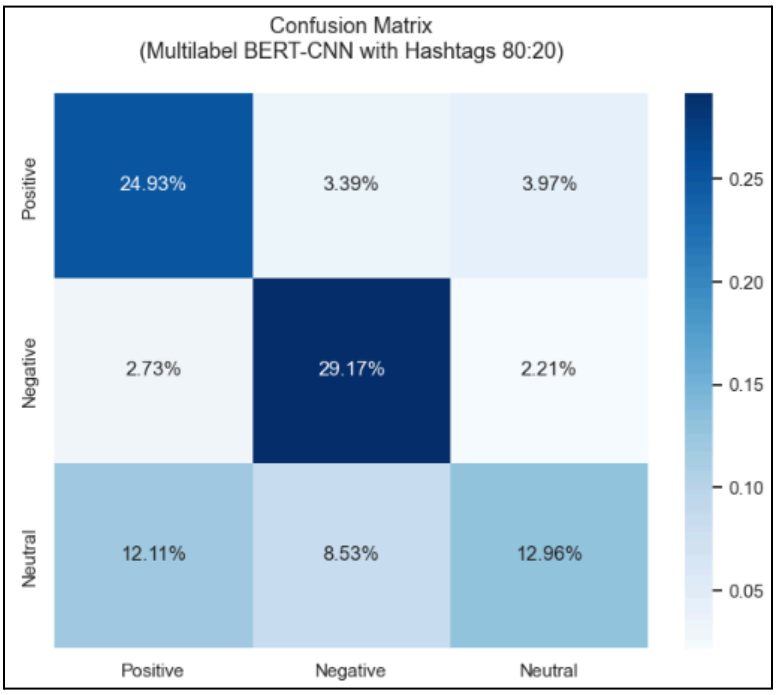
```
Classification Report:

              precision    recall  f1-score   support

    Positive     0.6268    0.7722    0.6920       496
    Negative     0.7100    0.8550    0.7758       524
     Neutral     0.6769    0.3857    0.4914       516

    accuracy                         0.6706      1536
   macro avg     0.6712    0.6709    0.6530      1536
weighted avg     0.6720    0.6706    0.6532      1536
```

Figure 4.84. Classification Report (BERT-CNN w/ HT 80:20)



383	52	61
42	448	34
186	131	199

Figure 4.85. Confusion Matrix (BERT-CNN w/ HT 80:20)

The split ratio for this model achieved the best results at epoch 4. At its lowest test loss, the model produced an output with 67.06% accuracy, 67.12% precision, 67.09% recall and 65.30% F1-Score. For the evaluation metrics, the model achieved the highest result with precision at 80:20 split ratio.

E.2.2.3. 90:10 Split

```
Epoch: 04

        Train Loss: 0.634
        Test Loss: 0.681

        Accuracy: 0.6914 | F1-Score: 0.6741
        Precision: 0.6844 | Recall: 0.6840
```

Figure 4.86. Performance Measures (BERT-CNN w/ HT 90:10)

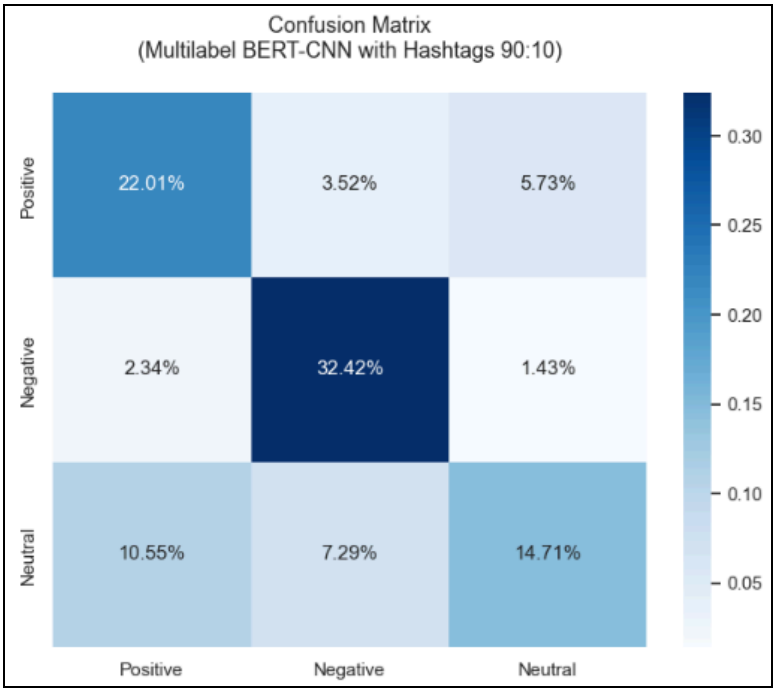
```
Classification Report:

              precision    recall  f1-score   support

    Positive     0.6306    0.7042    0.6654       240
    Negative     0.7500    0.8957    0.8164       278
     Neutral     0.6726    0.4520    0.5407       250

    accuracy                         0.6914       768
   macro avg     0.6844    0.6840    0.6741       768
weighted avg     0.6875    0.6914    0.6794       768
```

Figure 4.87. Classification Report (BERT-CNN w/ HT 90:10)



169	27	44
18	249	11
81	56	113

Figure 4.88. Confusion Matrix (BERT-CNN w/ HT 90:10)

The split ratio for this model achieved the best results at epoch 4. At its lowest test loss, the model produced an output with 69.14% accuracy, 68.44% precision, 68.40% recall and 67.41% F1-Score. Across all the evaluation metrics, the model achieved the highest result with accuracy at 90:10 split ratio.

E.4.3. BERT-CNN with ALL-CAPS

The BERT-CNN model, utilizing the cased version of BERT, incorporated the feature of retaining all capitalized words and phrases during sentiment analysis and classification of posts. It is speculated that ALL-CAPS may also hold meaningful context

and underlying context when used with intent to express. The model's reports will show how it improved over the base model.

E.4.3.1. 70-30 Split

```
Epoch: 05

        Train Loss: 0.644
        Test Loss: 0.759

        Accuracy: 0.6471 | F1-Score: 0.6342
        Precision: 0.6405 | Recall: 0.6465
```

Figure 4.89. Performance Measures (BERT-CNN w/ AC 70:30)

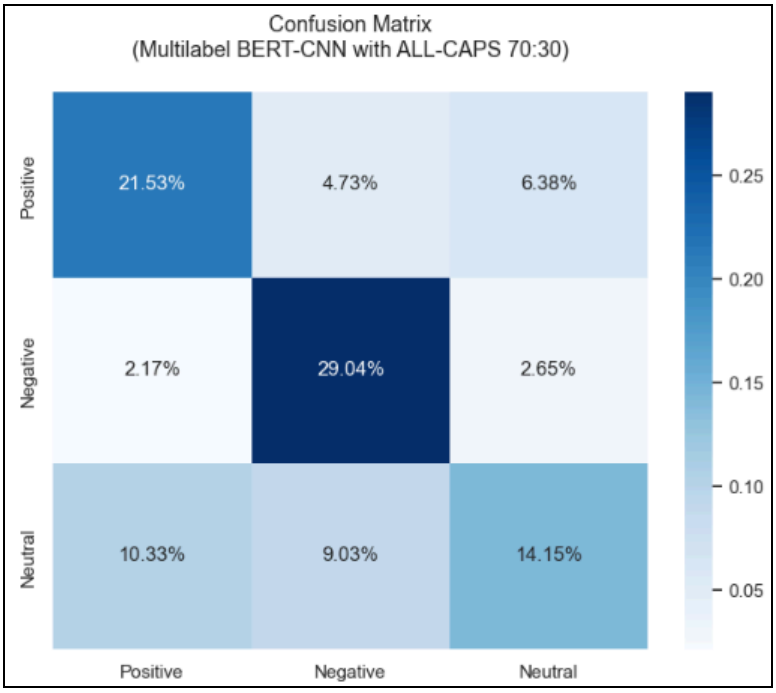
```
Classification Report:

              precision    recall  f1-score   support

    Positive     0.6327    0.6596    0.6458       752
    Negative     0.6785    0.8577    0.7576       780
     Neutral     0.6105    0.4223    0.4992       772

    accuracy                         0.6471      2304
   macro avg     0.6405    0.6465    0.6342      2304
weighted avg     0.6407    0.6471    0.6346      2304
```

Figure 4.90. Classification Report (BERT-CNN w/ AC 70:30)



496	109	147
50	669	61
238	208	326

Figure 4.91. Confusion Matrix (BERT-CNN w/ AC 70:30)

The split ratio for this model achieved the best results at the fifth epoch. At its lowest test loss, the model produced an output with 64.71% accuracy, 64.05% precision, 64.65% recall and 63.42% F1-Score. For all of the metrics, the model achieved the highest result with accuracy at 70:30 split ratio.

E.4.3.2. 80-20 Split

```
Epoch: 04

        Train Loss: 0.724
        Test Loss: 0.751

        Accuracy: 0.6504 | F1-Score: 0.6395
        Precision: 0.6408 | Recall: 0.6489
```

Figure 4.92. Performance Measures (BERT-CNN w/ AC 80:20)

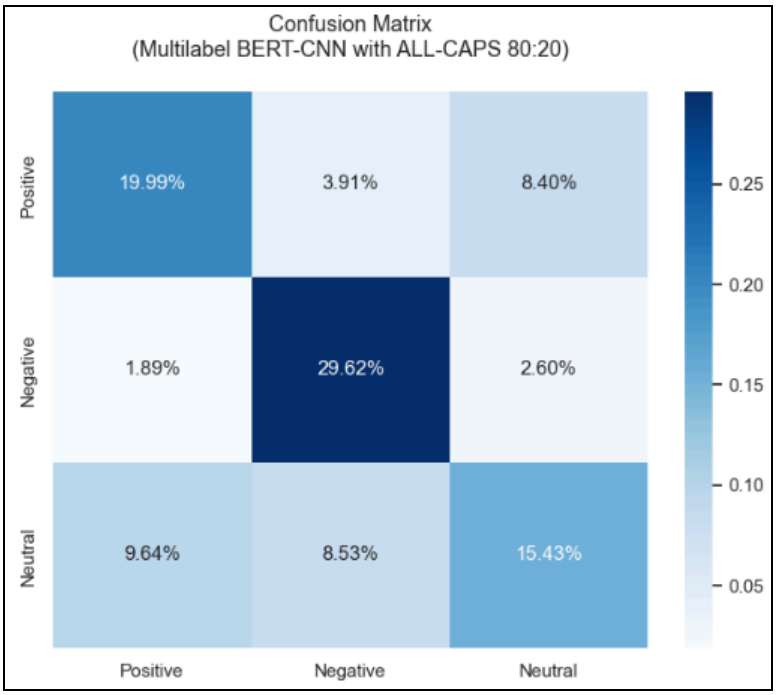
```
Classification Report:

              precision    recall  f1-score   support

    Positive     0.6343    0.6190    0.6265       496
    Negative     0.7043    0.8683    0.7778       524
     Neutral     0.5837    0.4593    0.5141       516

    accuracy                         0.6504      1536
   macro avg     0.6408    0.6489    0.6395      1536
weighted avg     0.6412    0.6504    0.6404      1536
```

Figure 4.93. Classification Report (BERT-CNN w/ AC 80:20)



307	60	129
29	455	40
148	131	237

Figure 4.94. Confusion Matrix (BERT-CNN w/ AC 80:20)

The split ratio for this model this time achieved the best results at the fourth epoch. At its lowest test loss, the model produced an output with 65.04% accuracy, 64.08% precision, 64.89% recall and 63.95% F1-Score. For the metrics, the model achieved the highest result with accuracy at 80:20 split ratio.

E.4.3.3. 90-10 Split

```
Epoch: 05

        Train Loss: 0.613
        Test Loss: 0.739

        Accuracy: 0.6771 | F1-Score: 0.6561
        Precision: 0.6756 | Recall: 0.6723
```

Figure 4.95. Performance Measures (BERT-CNN w/ AC 90:10)

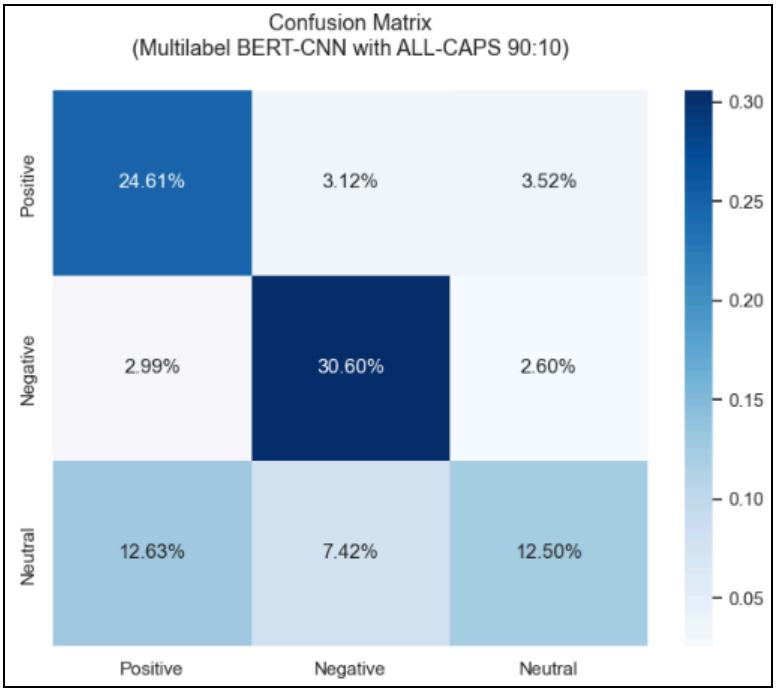
```
Classification Report:

              precision    recall  f1-score   support

    Positive     0.6117    0.7875    0.6885       240
    Negative     0.7437    0.8453    0.7912       278
     Neutral     0.6713    0.3840    0.4885       250

    accuracy                         0.6771       768
   macro avg     0.6756    0.6723    0.6561       768
weighted avg     0.6789    0.6771    0.6606       768
```

Figure 4.96. Classification Report (BERT-CNN w/ AC 90:10)



189	24	27
23	235	20
97	57	96

Figure 4.97. Classification Report (BERT-CNN w/ AC 90:10)

The split ratio for this model achieved the best results at the fifth epoch. At its lowest test loss this time, the model produced an output with 67.71% accuracy, 67.56% precision, 67.23% recall and 65.61% F1-Score. For the metrics, there is no doubt that the model achieved the highest result with recall at 90:10 split ratio.

E.4.4. BERT-CNN with Hashtags and ALL-CAPS

The final variant of the BERT-CNN model had the inclusion of HT and AC as a combination of features for the model's sentimental analysis and validation. Utilizing the cased version of BERT, this final configuration aimed to capture nuanced expressions and contextual subtleties conveyed by both hashtags and ALL-CAPS.

E.4.4.1. 70-30 Split

```
Epoch: 05

        Train Loss: 0.641
        Test Loss: 0.727

        Accuracy: 0.6645 | F1-Score: 0.6549
        Precision: 0.6598 | Recall: 0.6636
```

Figure 4.98. Performance Measures (BERT-CNN w/ HT+AC 70:30)

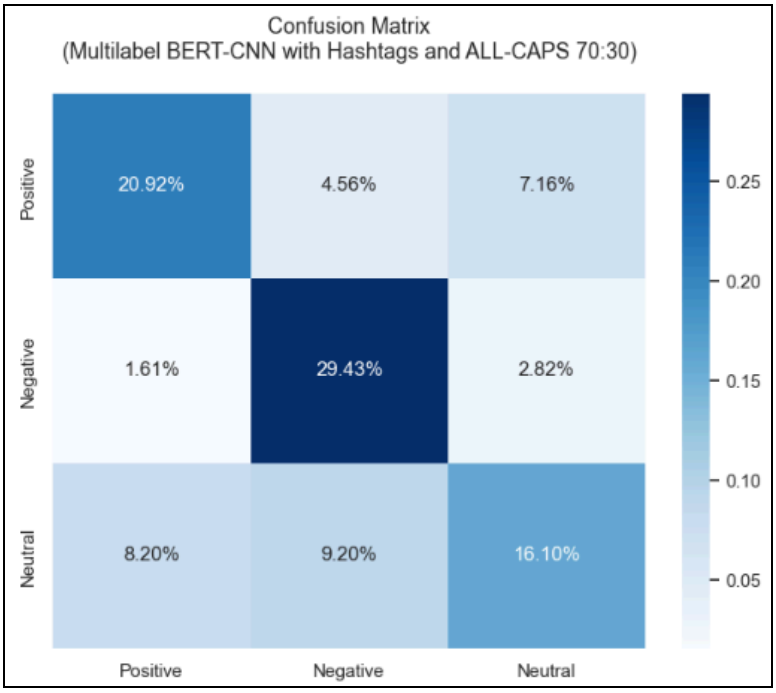
```
Classification Report:

              precision    recall  f1-score   support

    Positive     0.6808    0.6410    0.6603       752
    Negative     0.6814    0.8692    0.7639       780
     Neutral     0.6173    0.4806    0.5404       772

    accuracy                         0.6645      2304
   macro avg     0.6598    0.6636    0.6549      2304
weighted avg     0.6597    0.6645    0.6552      2304
```

Figure 4.99. Classification Report (BERT-CNN w/ HT+AC 70:30)



678	65	37
212	371	189
105	165	482

Figure 4.100. Confusion Matrix (BERT-CNN w/ HT+AC 70:30)

The split ratio for this model this time achieved the best results at epoch 5. At its lowest test loss, the model produced an output with 66.45% accuracy, 65.98% precision, 66.36% recall and 65.49% F1-Score. For the metrics, the model achieved the highest result with accuracy at 70:30 split ratio.

E.4.4.2. 80-20 Split

```
Epoch: 04

        Train Loss: 0.727
        Test Loss: 0.737

        Accuracy: 0.6589 | F1-Score: 0.6496
        Precision: 0.6515 | Recall: 0.6581
```

Figure 4.101. Performance Measures (BERT-CNN w/ HT+AC 80:20)

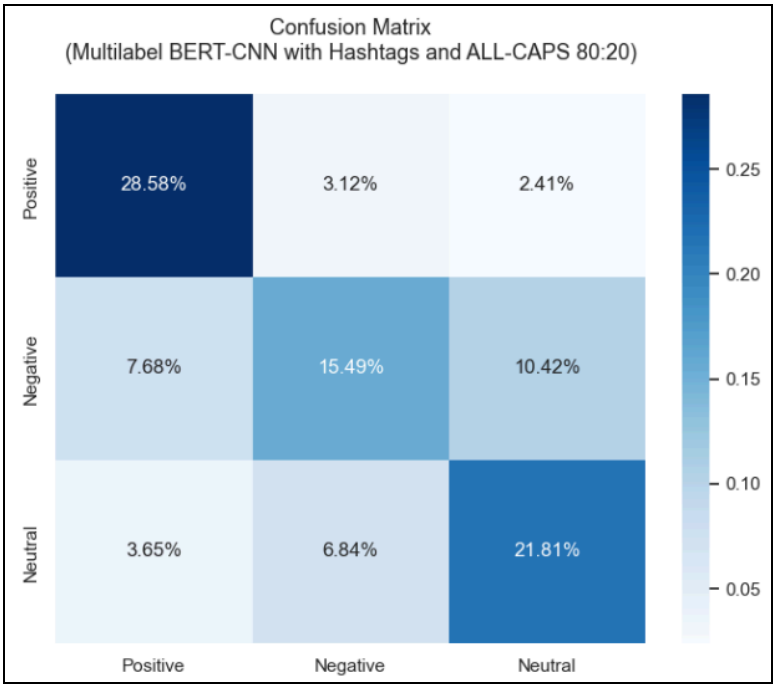
```
Classification Report:

              precision    recall  f1-score   support

    Positive     0.7162    0.8378    0.7722       524
    Negative     0.6087    0.4612    0.5248       516
     Neutral     0.6297    0.6754    0.6518       496

    accuracy                         0.6589      1536
   macro avg     0.6515    0.6581    0.6496      1536
weighted avg     0.6521    0.6589    0.6502      1536
```

Figure 4.102. Classification Report (BERT-CNN w/ HT+AC 80:20)



439	48	37
118	238	160
56	105	335

Figure 4.103. Confusion Matrix (BERT-CNN w/ HT+AC 80:20)

The split ratio for this model achieved the best results returning to the fourth epoch. At its lowest test loss, the model produced an output with 65.89% accuracy, 65.15% precision, 65.81% recall and 64.96% F1-Score. For the metrics, the model achieved the highest result with accuracy at 80:20 split ratio.

E.4.4.3. 90-10 Split

```
Epoch: 03

        Train Loss: 0.850
        Test Loss: 0.724

        Accuracy: 0.6510 | F1-Score: 0.6174
        Precision: 0.6537 | Recall: 0.6438
```

Figure 4.104. Performance Measures (BERT-CNN w/ HT+AC 90:10)

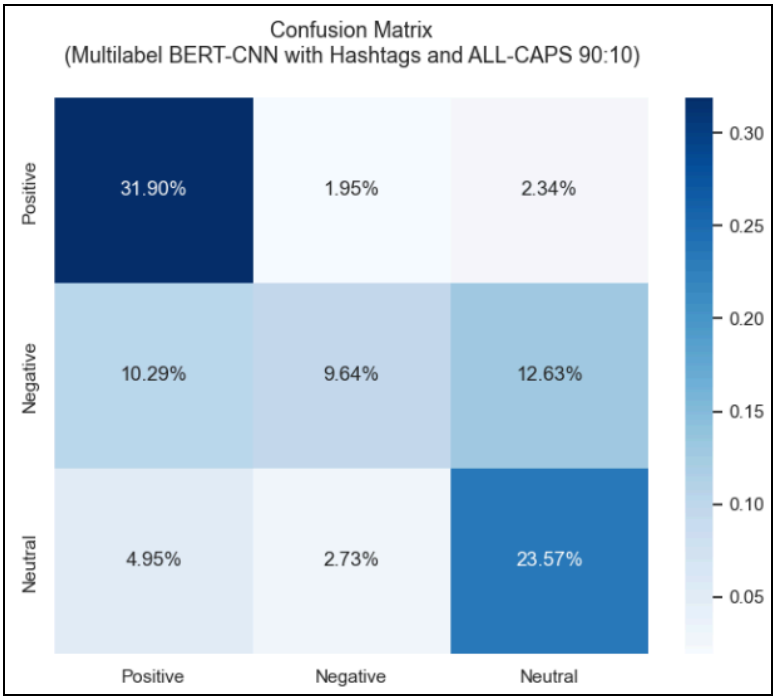
```
Classification Report:

              precision    recall  f1-score   support

    Positive     0.6768    0.8813    0.7656       278
    Negative     0.6727    0.2960    0.4111       250
     Neutral     0.6115    0.7542    0.6754       240

    accuracy                         0.6510       768
   macro avg     0.6537    0.6438    0.6174       768
weighted avg     0.6551    0.6510    0.6220       768
```

Figure 4.106. Classification Report (BERT-CNN w/ HT+AC 90:10)



245	15	18
79	74	97
38	21	181

Figure 4.107. Confusion Matrix (BERT-CNN w/ HT+AC 90:10)

The final split ratio for this model this time achieved the best results at the third epoch, which is the shortest epoch run for multi label. At its lowest test loss, the model produced an output with 65.10% accuracy, 65.37% precision, 64.38% recall and 61.74% F1-Score. For the metrics, the model achieved the highest result with accuracy at 80:20 split ratio.

E.5. Text Analysis with BERT Embeddings Integration

The BERT-CNN models heavily depended on contextual data, identifying specific words or phrases as conversational cues with significant embedded value for classifying posts as hate and non-hate. This section of the study delved into the exploration and discussion of textual data, specifically integrating BERT embeddings into the BERT-CNN model. The analysis aimed to elucidate the impact of BERT embeddings on the model, providing insights into visualizing the importance and value of the textual data on each post, especially concerning their classification as either hate or non-hate.

E.5.1 Candidate Names

The use of candidate names was important to the model as it held a large word embedding value from the dataset, influencing the model's predictions. Many posts gathered from X during the latest Philippine presidential elections mention candidate names as an approach to gain followers by expressing their thoughts or criticizing other opposing opinions. Due to this, candidate names became especially linked to classifying a post as hate and non-hate. Table 4.10. below provided a detailed list of candidate names along with their corresponding prediction values derived from the BERT-CNN model, which incorporated hashtags and ALL-CAPS. This table offered insights into how the model attributed labels based on the presence of candidate names, shedding light on the nuanced dynamics of hate speech detection within the context of the Philippine presidential elections.

Table 4.10. Sample Candidate Names with Prediction Values

Candidate Names	Prediction Value	Label
BBM	0.38	Non-Hate
bongbong	**0.91**	**Hate**
Marcos	**0.68**	**Hate**
Leni	0.19	Non-Hate
Robredo	0.19	Non-Hate
Isko	0.38	Non-Hate
moreno	0.298	Non-Hate
domagoso	0.15	Non-Hate
manny	0.23	Non-Hate
pacquiao	0.13	Non-Hate
ping	0.27	Non-Hate
lacson	0.40	Non-Hate
enrile	0.34	Non-Hate
abella	0.37	Non-Hate
faisal	0.43	Non-Hate
mangondato	0.28	Non-Hate

The candidate names showed a strong conversational cue that became a large contributor to the prediction's classification label. The researchers speculated this to be the cause of proper nouns and names being closely tied to the frequency of hate and non-hate. Such was the case for the word *Marcos* which was mostly tied to posts that were labeled as Hate. However, upon checking the dataset, it has been observed that the

amount of posts labeled as ‘Hate’ that contain the name of Bong Bong Marcos are 1263 is significantly higher compared to those that are labeled as ‘Non-hate’ which are 393. On the other hand, the amount of posts that contain the name of Leni Robredo that are labeled as ‘Hate’ amounts to 1388, higher than the ones that contain the name of Marcos, while the posts that are labeled as ‘Non-hate’ amounts to 1945. Hence, it can be said that the large value of Marcos’ name can be attributed to the number of labeled posts that contain his name with a relatively small number of posts labeled as Non-hate. Although the proportions are relatively the same for both example candidates, it can be said that the model had a small amount of Non-hate posts that pertain to Marcos to work with, hence the embedded value. This is compared to posts that pertain to Robredo where both Hate and Non-hate posts are at least 1000. This showed that names, specifically those with relation to the presidential election, hold a large embedding value on the model’s classification and emphasizing the need for careful consideration in the classification process.

E.5.2. Predictions

The BERT-CNN system models learned the process of classification through the process of training with labeled posts. The model then identified the most frequent words or cues and used them as indicators in predicting the newer posts from the test dataset. For example, the words "bongbong" and "Marcos" are frequently found on posts which were labeled as Hate while there is a lack of posts that contain the same words that are labeled as Non-hate. Hence, the model placed these keywords with a higher word embedding value which often led to Hate predictions. While this strengthens the

BERT-CNN model's ability to recall context clues in texts, it may also lead to misclassifications in predicting Hate and Non-Hate posts.

E.5.3. Misclassification

Misclassification was the result of wrongfully assigning someone or something to a category. This section of the study discussed the factors and mislabeled predictions of the BERT-CNN models along with the researchers' remarks.

Table 4.11. Misclassifications

Text Data	Actual	Predicted	Remarks
# BBMFORPRESIDENT let bong bong marcos make change in to our country keep seeking what is the right for our fellow filipino's and to the country marcos is the best president that we will ever had # votewisely	Non-Hate	Hate	As mentioned, this was an example of misclassification due the the large weight of candidate names, especially the name of Bongbong Marcos. Even though there were some instances where the model could classify Non-Hate posts containing Marcos' name, it tended to struggle with Non-Hate posts with the same context due to the large number of hate posts containing Marcos' name in the training set without the same number of non-hate posts .

Don’t want to talk ill of someone who obviously means well but Faisal Mangodato just simply talks too slowly #Pilipinasdebates	Hate	Non-Hate	This was another case of misclassification due the word value of a candidate’s name. This time, Faisal Mangodato is the candidate in question and the embedding value of his name was significantly different compared to Marcos, hence the Non-Hate misclassification

Provided on the table above were examples of misclassifications from the BERT-CNN model’s prediction. This proved that the model’s learning heavily relied on its training dataset and used keywords as cues in predicting hate or non-hate posts. At the same time, there were also words whose embedding may lean towards either Hate or non-hate. For example, the word “Lugaw” is classified as Non-hate despite it being a type of food. The reason for this, and the value of embedded text of candidate names can be explained by Exxact (2021) where it is stated that BERT embeddings are dependent on the context of its surrounding words. This is also backed up by Shaikh (2023) where the author mentioned that BERT’s self-attention mechanism in its architecture allows it to embed individual words based on its context. Hence, the embedding of ‘Marcos’, ‘Bongbong’ and ‘Lugaw’ can be attributed to the context in which the words are used which are usually words that describe Marcos like ‘Dictator’, or hate speech towards Leni Robredo for ‘Lugaw’. This proved that the BERT word embeddings approach

served as an advantage to the model's performance, but it also brings a risk to misclassifying text based on the embeddings of a certain word, namely the names of the campaigned presidential candidates against the whole context.

E.6. Model Comparisons

The team conducted a study with four variations of BERT-CNN paired with a combination of features such as Hashtags and ALL-CAPS. This section of the study compared the models' performance measures namely, *BERT-CNN and fastText CNN, BERT-CNN with Hashtags and BERT-CNN without Hashtags, BERT-CNN with ALL-CAPS and BERT-CNN without ALL-CAPS,* as well as *BERT-CNN with Hashtags and ALL-CAPS and BERT-CNN without Hashtags and ALL-CAPS.*

E.6.1. BERT-CNN and fastText CNN Comparison

Table 4.12 Comparison of Performance Measures for fastText CNN and BERT-CNN models for Binary Classification

Split	Accuracy		Precision		Recall		F1-Score	
	BERT-CNN	fastText CNN	BERT-CNN	fastText CNN	BERT-CNN	fastText CNN	BERT-CNN	fastText CNN
70:30	**90.30%**	85.61%	**90.30%**	85.97%	**90.30%**	85.70%	**90.30%**	85.59%
80:20	**90.62%**	85.94%	**90.64%**	86.02%	**90.63%**	85.92%	**90.62%**	85.92%
90:10	**93.55%**	86.52%	**93.59%**	86.85%	**93.58%**	86.47%	**93.55%**	86.49%

The comparison between the BERT-CNN and *fastText* CNN models for binary classification revealed notable differences in performance across various train-test splits.

BERT-CNN consistently outperformed *fastText* CNN in terms of accuracy, precision, recall, and F1-score. In the 70:30 split, BERT-CNN achieved an accuracy of 90.30%, while *fastText* CNN lags behind at 85.61%. This trend continued across splits, with BERT-CNN maintaining higher precision and recall values. The 90:10 split showcased a substantial improvement in BERT-CNN's performance, with an accuracy of 93.55% and a remarkable recall of 93.59%. In contrast, *fastText* CNN exhibited more modest gains as the training set size increased. Overall, BERT-CNN demonstrated a clear advantage over *fastText* CNN, highlighting its superior predictive capabilities and suitability for binary classification tasks across different train-test splits.

E.6.2. BERT-CNN w/ HT and BERT-CNN w/o HT Comparison

Table 4.13 Comparison of Performance Measures for BERT-CNN with Hashtags and BERT-CNN without Hashtags models for Binary Classification

Split	Accuracy		Precision		Recall		F1-Score	
	BERT-CNN w/ HT	BERT-CNN w/o HT	BERT-CNN w/ HT	BERT-CNN w/o HT	BERT-CNN w/ HT	BERT-CNN w/o HT	BERT-CNN w/ HT	BERT-CNN w/o HT
70:30	**90.69%**	90.30%	**90.69%**	90.30%	**90.69%**	90.30%	**90.69%**	90.30%
80:20	90.14%	**90.62%**	90.16%	**90.64%**	90.15%	**90.63%**	90.14%	**90.62%**
90:10	92.38%	**93.55%**	91.76%	**93.59%**	92.86%	**93.58%**	92.31%	**93.55%**

The comparison between BERT-CNN models with and without hashtags (HT) for binary classification highlighted subtle variations in their performance metrics. In the 70:30 split, the BERT-CNN with HT achieved an accuracy of 90.69%, precision of 90.69%, recall of 90.33%, and an F1-score of 90.51%, surpassing the corresponding

metrics of the model without HT by a small margin. However, BERT-CNN without HT consistently exhibited slightly higher accuracy, precision, recall, and F1-score values in the 80:20 and 90:10 splits. This trend showcased a consistent advantage for the model without HT. While both models demonstrated strong binary classification capabilities, the absence of hashtags appeared to contribute to a modest but consistent improvement in performance metrics for BERT-CNN.

E.6.3. BERT-CNN w/ AC and BERT-CNN w/o AC Comparison

Table 4.14 Comparison of Performance Measures for BERT-CNN with ALL-CAPS and BERT-CNN without ALL-CAPS models for Binary Classification

Split	Accuracy		Precision		Recall		F1-Score	
	BERT-CNN w/ AC (Cased)	BERT-CNN w/o AC (Uncased)	BERT-CNN w/ AC (Cased)	BERT-CNN w/o AC (Uncased)	BERT-CNN w/ AC (Cased)	BERT-CNN w/o AC (Uncased)	BERT-CNN w/ AC (Cased)	BERT-CNN w/o AC (Uncased)
70:30	89.58%	**90.30%**	88.49%	**90.30%**	**90.60%**	90.30%	89.53%	**90.30%**
80:20	88.48%	**90.62%**	87.64%	**90.64%**	89.37%	**90.63%**	88.50%	**90.62%**
90:10	91.41%	**93.55%**	90.00%	**93.59%**	92.86%	**93.58%**	91.41%	**93.55%**

The comparison of performance measures for BERT-CNN with and without ALL-CAPS (AC) in binary classification revealed notable distinctions. The uncased BERT-CNN, without AC, consistently outperformed its cased counterpart with AC across all splits, demonstrating superior accuracy, precision, recall, and F1-score—except for the 70:30 split, where BERT-CNN with AC exhibited higher recall at 90.60%. Specifically, at the 90:10 split, the uncased model achieved an impressive accuracy of 93.55%, surpassing the cased model's 91.41%. This suggests that the uncased BERT-CNN model

exhibited greater proficiency in handling the classification task. Notably, a study by Dao and Aizawa (2023) found that the uncased BERT model displayed enhanced resilience to inconsistencies in capitalization within noisy text data, whereas the BERT-base-cased model excelled in well-written text with clearly provided case information. The observed performance gap in this study's models may stem from the heightened sensitivity of the cased model to noise and inconsistencies, or it could be influenced by the prevalence of fewer ALL-CAPS instances in the dataset.

E.6.4. BERT-CNN w/ HT+AC and BERT-CNN w/o HT+AC Comparison

Table 4.15 Comparison of Performance Measures for BERT-CNN with Hashtags and ALL-CAPS and BERT-CNN without Hashtags and ALL-CAPS models for Binary Classification

Split	**Accuracy**		**Precision**		**Recall**		**F1-Score**	
	BERT-CNN w/ HT and AC (Cased)	BERT-CNN w/o HT and AC (Uncased)	BERT-CNN w/ HT and AC (Cased)	BERT-CNN w/o HT and AC (Uncased)	BERT-CNN w/ HT and AC (Cased)	BERT-CNN w/o HT and AC (Uncased)	BERT-CNN w/ HT and AC (Cased)	BERT-CNN w/o HT and AC (Uncased)
70:30	89.39%	**90.30%**	87.95%	**90.30%**	**90.86%**	90.30%	89.38%	**90.30%**
80:20	**90.82%**	90.62%	**90.75%**	90.64%	**90.75%**	90.63%	**90.75%**	90.62%
90:10	91.99%	**93.55%**	90.73%	**93.59%**	93.25%	**93.58%**	91.98%	**93.55%**

The uncased BERT-CNN without HT and AC consistently outperformed its cased counterpart across all splits, exhibiting higher accuracy, precision, recall, and F1-score—except for the 80:20 split, where cased BERT-CNN with HT+AC exhibited higher accuracy, precision and F1-score, and for the 70:30 split with higher recall. The

performance of the cased model, while competitive, showed less variability across different splits, with minimal improvements in precision and recall. These findings emphasize the impact of preprocessing choices, such as the inclusion of hashtags and ALL-CAPS, on model performance and underline the effectiveness of the uncased BERT-CNN configuration in this particular binary classification scenario.

E.6.5. BERT-CNN and fastText CNN Comparison

Table 4.16 Comparison of Performance Measures for fastText CNN and BERT-CNN models for Multi Label Classification

Split	Accuracy		Precision		Recall		F1-Score	
	BERT-CNN	fastText CNN	BERT-CNN	fastText CNN	BERT-CNN	fastText CNN	BERT-CNN	fastText CNN
70:30	**65.54%**	62.63%	**65.46%**	61.97%	**65.52%**	62.54%	**63.21%**	61.82%
80:20	**66.28%**	62.89%	**65.82%**	62.11%	**66.24%**	62.79%	**64.49%**	62.07%
90:10	**67.84%**	63.80%	**66.60%**	63.09%	**67.10%**	63.41%	**66.27%**	62.74%

The results presented in Table 4.15 highlighted the superior performance of the BERT-CNN model in comparison to the *fastText* CNN model across various split ratios (70:30, 80:20, and 90:10). The evaluation metrics, including accuracy, precision, recall, and F1-score, consistently demonstrated higher values for the BERT-CNN model compared to the base study's model. Specifically, the BERT-CNN model exhibited a notable superiority in accuracy, surpassing the *fastTex*t CNN model by 2.91% absolute accuracy on the 70:30 train-test split, 3.39% on the 80:20 train-test split, and 4.01% on

the 90:10 train-test split. These findings suggested that the BERT-CNN model outperforms the *fastTex*t CNN model across different data split ratios, emphasizing its effectiveness in the context of the study.

E.6.6. BERT-CNN w/ HT and BERT-CNN w/o HT Comparison

Table 4.17 Comparison of Performance Measures for BERT-CNN with Hashtags and BERT-CNN without Hashtags models for Multi Label Classification

Split	Accuracy		Precision		Recall		F1-Score	
	BERT-CNN w/ HT	BERT-CNN w/o HT	BERT-CNN w/ HT	BERT-CNN w/o HT	BERT-CNN w/ HT	BERT-CNN w/o HT	BERT-CNN w/ HT	BERT-CNN w/o HT
70:30	**67.93%**	65.54%	**67.37%**	65.46%	**67.90%**	65.52%	**66.90%**	63.21%
80:20	**67.06%**	66.28%	**67.12%**	65.82%	**67.09%**	66.24%	**65.30%**	64.49%
90:10	**69.14%**	67.84%	**68.44%**	66.60%	**68.40%**	67.10%	**67.41%**	66.27%

The significance of incorporating HT in multi label classification, specifically distinguishing between positive, negative, and neutral sentiments, was evident in the performance of the BERT-CNN model. When hashtags were included, the model demonstrated a slight but consistent improvement across all evaluation metrics and for all split ratios compared to the model without HT. Furthermore, the BERT-CNN model, even with hashtags, outperformed the fastText CNN model across all metrics for every train-test split. The BERT-CNN model with HT achieved the highest absolute accuracy, surpassing the model without HT by 1.23% on the 70:30 split ratio, 0.78% on the 80:20 split, and 0.33% on the 90:10 split. These results highlighted the positive impact of incorporating hashtags in the multi label classification task, indicating that the inclusion of this contextual information contributed to the model's ability to accurately classify

sentiments. The consistent outperformance of the BERT-CNN model with HT, especially in terms of accuracy, reinforces the relevance and effectiveness of leveraging hashtag information for improved sentiment analysis in multi label scenarios.

E.6.7. BERT-CNN w/ AC and BERT-CNN w/o AC Comparison

Table 4.18 Comparison of Performance Measures for BERT-CNN with ALL-CAPS and BERT-CNN without ALL-CAPS models for Multi Label Classification

Split	Accuracy		Precision		Recall		F1-Score	
	BERT-CNN w/ AC (Cased)	BERT-CNN w/o AC (Uncased)	BERT-CNN w/ AC (Cased)	BERT-CNN w/o AC (Uncased)	BERT-CNN w/ AC (Cased)	BERT-CNN w/o AC (Uncased)	BERT-CNN w/ AC (Cased)	BERT-CNN w/o AC (Uncased)
70:30	64.71%	**65.54%**	64.05%	**65.46%**	64.65%	**65.52%**	**63.42%**	63.21%
80:20	65.04%	**66.28%**	64.08%	**65.82%**	64.89%	**66.24%**	63.95%	**64.49%**
90:10	67.71%	**67.84%**	**67.56%**	66.60%	**67.23%**	67.10%	65.61%	**66.27%**

The evaluation of the BERT-CNN model as an uncased model without the AC indicated that its performance was marginally superior when compared to the cased BERT-CNN model with AC. This difference, however, was relatively small and inconsistent across various metrics. Despite the overall slightly better performance without AC, the BERT-CNN model with AC demonstrated specific improvements. Specifically, BERT-CNN with AC outperformed the model without AC by 0.21% in F1-Score on the 70:30 split ratio. Additionally, there were gains of 0.96% in precision and 0.13% in recall on the 90:10 split for the model with AC. These findings suggest that while the uncased model without AC showed a slight edge in overall performance, the

inclusion of AC contributed to specific enhancements in precision, recall, and F1-Score, particularly in certain split ratios.

E.6.8. BERT-CNN w/ HT+AC and BERT-CNN w/o HT+AC Comparison

Table 4.19 Comparison of Performance Measures for BERT-CNN with Hashtags and ALL-CAPS and BERT-CNN without Hashtags and ALL-CAPS models for Multi Label Classification

Split	Accuracy		Precision		Recall		F1-Score	
	BERT-CNN w/ HT and AC (Cased)	BERT-CNN w/o HT and AC (Uncased)	BERT-CNN w/ HT and AC (Cased)	BERT-CNN w/o HT and AC (Uncased)	BERT-CNN w/ HT and AC (Cased)	BERT-CNN w/o HT and AC (Uncased)	BERT-CNN w/ HT and AC (Cased)	BERT-CNN w/o HT and AC (Uncased)
70:30	**66.45%**	65.54%	**65.98%**	65.46%	**66.36%**	65.52%	**65.49%**	63.21%
80:20	65.89%	**66.28%**	65.15%	**65.82%**	65.81%	**66.24%**	**64.96%**	64.49%
90:10	65.10%	**67.84%**	65.37%	**66.60%**	64.38%	**67.10%**	61.74%	**66.27%**

The uncased BERT-CNN without HT and AC demonstrated superior overall performance compared to its cased counterpart with HT and AC. While the cased model exhibited superiority in the 70:30 split across all evaluation metrics, including a notable improvement in F1-score in the subsequent split, the overall trend showed a decline and inconsistency in other metrics.

Although the cased model displayed a brief advantage in specific split ratios, it was overshadowed by the uncased model's sustained and more consistent superiority in performance. This emphasized the nuanced impact of hashtags and ALL-CAPS inclusion

on model performance, suggesting that the uncased BERT-CNN without HT and AC may offer more robust and reliable performance across a broader range of scenarios.

E.7. Statistical Analysis

This section of the study conducted the hypothesis testing whether the performance of the models will result in either accepting or rejecting the null hypothesis.

Table 4.20. McNemar's Test with Holm-Bonferroni for BERT-CNN and fastText CNN models for Binary Label Classification

Split	**Hypothesis Testing (α = 0.05)**		
	p-value	**HB correction**	**Result**
70:30	0.0000001871	0.016	Reject Null Hypothesis
80:20	0.0000537307	0.05	Reject Null Hypothesis
90:10	0.0000048175	0.025	Reject Null Hypothesis

Table 4.21. McNemar's Test with Holm-Bonferroni for BERT-CNN and fastText CNN models for Multi Label Classification

Split	Hypothesis Testing (α = 0.05)		
	p-value	**HB correction**	**Result**
70:30	0.0064897136	0.016	Reject Null Hypothesis
80:20	0.0118929794	0.025	Reject Null Hypothesis
90:10	0.0386433487	0.05	Reject Null Hypothesis

Based on the results of a McNemar's test with Holm-Bonferroni correction for BERT-CNN and *fastText* CNN models for both binary and multi label classification, the BERT-CNN model outperformed the *fastTex*t CNN model for binary classification with the binary labeled dataset as well as the multi classification with the multi labeled dataset. The results on Table 4.20 have shown that there is a significant improvement in the performance between the BERT-CNN and *fastText* CNN models. This suggests that BERT can exceed fastText as a powerful language model to extract meaningful features as well as underlying context from posts.

F. Summary of Findings

This section of the chapter summarizes the insights and findings from the results of the researcher's experimentations, testings and analysis. The insights were organized into a bullet form showing the key understandings of utilizing the BERT-CNN approach in detecting hate or non hate speech in text.

- The system architecture, consisting of five connected modules namely, Preprocessing, Splitting, BERT Word Embeddings, Binary and Multi Label CNN, and Evaluation, was designed to detect hate speech from posts in context with the late 2022 Philippine presidential election-related posts.

- BERT-CNN models were trained with four variations: (1) BERT-CNN, (2) BERT-CNN with Hashtags, (3) BERT-CNN with ALL-CAPS, and (4) BERT-CNN with Hashtags and ALL-CAPS.

- The CNN model architecture was adaptable to both binary and multi label classification tasks, employing embedding layers, convolutional layers, batch normalization, pooling layers, dropout layers, and a fully connected layer with different configurations for binary and multi label outputs. The incorporation of **Embedding Output Layer** was able to slightly help control the model's performance from overfitting.

- BERT was reliant on contextual data unlike fastText that lacked contextual understanding. BERT-CNN model outperformed *fastText* CNN models in binary and multi label classification for hate speech detection.

- The pivotal improvement in BERT-CNN models stemmed from the innate ability of BERT to comprehend context. The inclusion of both **English and Filipino stopwords**, along with **candidate names**, played a crucial role in enhancing the performance of BERT-CNN models in both binary and multi-label classifications, deviating from the preprocessing approach employed in the base study. This highlighted BERT's exceptional capacity for nuanced contextual understanding.

- The utilization of BERT embeddings provided the model with a contextual lens, enabling it to grasp the intricate relationships between words and their meanings within the election-related discourse. Candidate names, especially those related to the presidential election, demonstrated high embedding value in classifying posts as hate or non-hate and positive, negative and neutral. This also resulted in misclassifications based on context and embedded values. In essence, the integration of BERT embeddings not only enhanced the model's overall performance but also illuminated the challenges associated with nuanced contextual analysis in hate speech detection.

- BERT-CNN consistently achieved a higher performance compared to the *fastText* CNN model across all split ratios, 70:30, 80:20, and 90:10.

- BERT-CNN with hashtags (HT) for binary label classification achieved comparable performance to BERT-CNN without HT, with a slight advantage in the 70:30 split. Contrary to the base study's recommendation, the team found that the absence of hashtags in text still contributed to a consistent improvement in BERT-CNN performance measures for binary classification. Contrarily, in the

case of multi-label classification, hashtags proved instrumental in effectively categorizing posts based on their respective sentiments.

- The uncased BERT-CNN model without retainment of ALL-CAPS (AC) in text consistently outperformed cased binary and multi label BERT-CNN with AC models across all splits, showing higher value in accuracy, precision, recall, and F1-score. There were only a few instances where BERT-CNN with AC achieved higher precision or recall. Despite the cased versions being more sensitive to data inconsistencies and noise, they still effectively managed to handle the presence of ALL-CAPS in the text.

- The uncased BERT-CNN model without retainment of both Hashtags and ALL-CAPS in text consistently outperformed the cased binary and multi label models across all splits, exhibiting higher accuracy, precision, recall, and F1-score. However, the BERT-CNN model with HT and AC was superior in the 80:20 split for binary classification and 70:30 split for multi label classification.

- McNemar's test with Holm-Bonferroni correction indicated a significant improvement in BERT-CNN model performance compared to the *fastText* CNN in binary and multi label classification.

Chapter V

Summary of Findings, Conclusions and Recommendations

This chapter encapsulated the summary of findings of the study, its conclusions and recommendations for future studies.

A. Summary

The goal of this study was to evaluate the performance of hate speech detection in Philippine election-related posts by implementing the BERT-CNN model as compared to the base study's CNN with *fastText* word embeddings. The study utilized a labeled dataset extracted from Arganosa et al.'s (2022) base study.

The system comprised five (5) major modules: Preprocessing, Splitting, BERT Word Embeddings, Binary and Multi Label CNN, and Evaluation. The Preprocessing module consisted of six (6) submodules, including data extraction, URL and account handle removal, special characters removal, hashtag processing, normalization, and tokenization. The Splitting module segmented the preprocessed data into training and testing sets, with variations in the proportion, 70/30, 80/20, 90/10 for different experiments. The BERT token embeddings was passed to two (2) types of CNN namely, Binary CNN and Multi Label CNN. The Evaluation module then tested the trained Binary CNN and Multi label CNN which resulted in two types of outputs: Classified posts as either *Hate, Non-Hate* or *Positive, Neutral, Negative.*

Aside from the BERT-CNN model, three variations of the BERT-CNN model were also tested: (1) BERT-CNN with Hashtags, (2) BERT-CNN with ALL-CAPS, and (3) BERT-CNN with Hashtags and ALL-CAPS. The following key findings emerged:

1. **BERT-CNN Model**

   a. ***Binary Classification (Hate or Non-Hate)***

      i. *BERT-CNN*

         - BERT-CNN consistently exhibited an impressive performance in accuracy, precision, recall, and F1-score across various train-test splits for binary classification. In the 70:30 split, BERT-CNN achieved an accuracy, precision, recall, and F1-score of 90.30%. Similarly, for the 80:20 split, the model maintained high metrics, achieving an accuracy of 90.23%, precision of 90.64%, recall of 90.63%, and an F1-score of 90.62%. In the 90:10 split, the model excelled further, attaining a remarkable accuracy of 93.55%, precision of 93.59%, recall of 93.58%, and an F1-score of 93.55%.

      ii. *BERT-CNN with vs. without Hashtags*

         - BERT-CNN with Hashtags showed a slight advantage in the 70:30 split with 90.69% in all performance measures, while BERT-CNN without Hashtags consistently exhibited slightly higher performance across other splits.

iii. BERT-CNN with vs. without ALL-CAPS

- Uncased BERT-CNN consistently outperformed its cased counterpart across all splits in binary classification except in the 70:30 split where the cased BERT-CNN with ALL-CAPS achieved a higher recall of 90.60%.
- The uncased model demonstrated greater proficiency in handling the classification task, suggesting sensitivity of the cased model to noise and inconsistencies.

iv. BERT-CNN with Hashtags and ALL-CAPS vs. without both

- The cased BERT-CNN model with Hashtags and ALL-CAPS outperformed the model without both in the 70:30 split with a recall of 90.86% and in the 80:20 split with 90.82% accuracy, and 90.75% for precision, recall, and F1-score. But the model without both was superior in the other performance measures within the same splits and in 90:10.

b. Multi Label Classification (Positive, Negative, or Neutral)

i. BERT-CNN

- BERT-CNN consistently demonstrated remarkable performance across different train-test splits in terms of accuracy, precision, recall, and F1-score

for multi label classification. In the 70:30 split, BERT-CNN delivered an impressive performance, boasting an accuracy of 65.54%, precision of 65.46%, recall of 65.52%, and an F1-score of 63.21%. Similarly, across the 80:20 split, the model maintained robust metrics, achieving an accuracy of 66.28%, precision of 65.82%, recall of 66.24%, and an F1-score of 64.49%. Further showcasing its prowess, in the 90:10 split, the model excelled with outstanding accuracy at 67.84%, precision of 66.60%, recall of 67.10%, and an F1-score of 66.27%.

ii. BERT-CNN with vs. without Hashtags

- The inclusion of hashtags in BERT-CNN for multi label classification led to consistent improvements across all performance measures and split ratios. In the 70:30 split, BERT-CNN showcased remarkable performance, boasting an accuracy of 67.93%, precision of 67.37%, recall of 67.90%, and an F1-score of 66.90%. Despite slightly lower metrics in the 80:20 split, the model still outperformed the model without Hashtags, achieving an accuracy of 67.06%, precision of 67.12%, recall of 67.09%, and

an F1-score of 65.30%. In the 90:10 split, the model continued to excel, achieving an outstanding accuracy of 69.14%, precision of 68.44%, recall of 68.40%, and an F1-score of 67.41%.

- Hashtags positively impacted accuracy, with the BERT-CNN model with hashtags outperforming the model without hashtags in each split.

iii. BERT-CNN with vs. without ALL-CAPS

- The uncased BERT-CNN without ALL-CAPS showed marginally superior performance overall, but the cased model with ALL-CAPS exhibited specific enhancements in precision, recall, and F1-score in certain split ratios. In the 70:30 split, the cased model attained a F1-score of 63.42%, while in the 90:10 split, it demonstrated a precision of 67.56% and a recall of 67.23%.

iv. BERT-CNN with Hashtags and ALL-CAPS vs. without both

- The cased BERT-CNN with Hashtags and ALL-CAPS was superior in the 70:30 split having an accuracy of 66.45%, precision of 65.98%, recall of 66.36%, and F1-score of 65.49%. The uncased BERT-CNN without Hashtags and ALL-CAPS demonstrated more robust and consistent

performance in the other splits compared to its cased counterpart.

2. BERT-CNN Model vs. fastText CNN Model

a. Binary Classification

- BERT-CNN consistently surpassed *fastText* CNN across a range of train-test splits, showcasing superior performance in terms of accuracy, precision, recall, and F1-score. BERT-CNN's performance measures ranged from 90% to 93%, while *fastText* CNN achieved around 85% to 86%. For the McNemar's Test with Holm-Bonferroni Correction, the null hypothesis was rejected in all splits indicating that there is a significant improvement in the performance between the BERT-CNN model and *fastText* CNN in the sentiment analysis of election-related posts into hate or non-hate classification.

b. Multi Label Classification

- BERT-CNN consistently outperformed *fastText* CNN in multi-label classification, demonstrating higher accuracy, precision, recall, and F1-score across various split ratios. BERT-CNN's performance measures consistently hovered around 63% to 67%, while *fastText* CNN achieved around 61% to 63%. In each split of the McNemar's Test with Holm-Bonferroni Correction, the null hypothesis was

consistently rejected. This rejection across all splits signified a significant improvement in performance when comparing the BERT-CNN model to the *fastText* CNN in the sentiment analysis of Philippine election-related posts.

3. **Text Analysis with Integration of BERT Embeddings**

   a. Candidate Names

   - The study focused on the integration of BERT embeddings into the BERT-CNN model for hate speech detection in the context of the Philippine presidential elections.
   - Candidate names played a crucial role in predictions, possessing a significant word embedding value.
   - Posts mentioning candidate names served as conversational cues, influencing the classification of posts as hate or non-hate.
   - Notable candidate names, such as Marcos and Leni Robredo, exhibited varying frequencies in hate and non-hate posts.
   - The study emphasized the need for careful consideration in classification due to the substantial influence of candidate names on the model's decisions.

   b. Predictions

   - The BERT-CNN models employed a learning process grounded in the integration of BERT embeddings, utilizing

training with labeled posts to identify frequent words or cues as indicators for predicting new posts in the test dataset.

- Keywords like "bongbong" and "Marcos" were identified as having higher word embedding values and were associated with hate predictions due to their prevalence in posts labeled as hate.
- While this approach enhanced the model's contextual understanding and recall of cues, it also posed a risk of misclassifications in predicting hate and non-hate posts.

c. Misclassifications

- Misclassification of the word "Lugaw" as non-hate, demonstrated the model's reliance on training data and keyword cues.
- The study attributed misclassifications to the context-dependent nature of BERT embeddings, where words like "Marcos" or "Lugaw" were embedded based on contextual usage.
- While acknowledging the advantage of BERT embeddings in improving performance, the study highlighted the risk of misclassifying text based on specific word embeddings, particularly the names of presidential candidates, without considering the broader context.

B. Conclusions

Having deployed the BERT-CNN model, this study achieved its primary goal of assessing the effectiveness of the BERT-CNN model in identifying political hate speech on X, demonstrating its proficiency in sentiment analysis of Filipino election-related posts. The comparison with the model proposed by Argañosa et al. (2022) consistently favored BERT-CNN, validating its superior performance. The following conclusions were derived based on the deployment of the presented system and summary of findings:

1. The BERT-CNN model demonstrated exceptional proficiency in classifying Philippine election-related posts into hate or non-hate categories, achieving high accuracy, precision, recall, and F1-score, each exceeding 90%. The consistency of these outstanding results across a spectrum of train-test splits highlighted the model's robust performance. Regardless of the variations introduced, such as the incorporation of Hashtags, ALL-CAPS, or both, the BERT-CNN model maintained its superior classification capabilities. These findings underscored the model's resilience and effectiveness in handling diverse textual data related to Philippine elections, making it a reliable and versatile tool for sentiment analysis in this context.
2. The comparison between the BERT-CNN and *fastText* CNN models revealed notable advancements in hate speech detection achieved by the BERT-CNN model. Demonstrating consistent superiority across diverse performance measures and splits, the BERT-CNN model exhibits a robust ability to classify effectively in various scenarios. The key strength of BERT-CNN lies in its capacity to capture intricate contextual relationships within language, rendering it particularly

well-suited for complex sentiment analysis tasks. The model's persistent outperformance implies its potential to offer more accurate and nuanced predictions when compared to the fastText CNN. Furthermore, the results of the McNemar's Test with Holm-Bonferroni correction affirmed a significant improvement in performance for the BERT-CNN model compared to the *fastText* CNN in the sentiment analysis of election-related posts, specifically in hate or non-hate classification, across all train-test splits.

3. The integration of BERT embeddings into the BERT-CNN model, significantly enhanced the model's performance in sentiment analysis of Philippine election-related posts. The study delved into the intricate dynamics of hate speech detection within the context of the Philippine presidential elections. One noteworthy aspect is the substantial influence of candidate names on the model's predictions, indicating a large word embedding value. Candidate names emerged as robust conversational cues, becoming integral to the classification of posts as hate or non-hate. This is exemplified by the detailed analysis of candidate names such as Marcos and Leni Robredo, showcasing their varying frequencies in hate and non-hate posts. The BERT-CNN model demonstrated its ability to learn from training data, identifying frequent words like "bongbong" and "Marcos" as indicators for hate predictions, thereby improving its contextual understanding. However, the study also uncovered potential misclassifications, emphasizing the model's reliance on training data and the context-dependent nature of BERT embeddings. Despite the risk of misclassification, the BERT word embeddings approach was deemed advantageous in enhancing the model's overall

performance, showcasing its ability to embed words based on contextual usage and providing valuable insights into the sentiment analysis of election-related posts.

C. Recommendations

Based on the findings of the study and its comprehensive conclusions, the researchers recommend the following :

1. Choice of Model or Further Hyperparameter Tuning: Although the created BERT-CNN model showed excellent results when it comes to Accuracy, Precision, Recall, and F1 score, it was noticed by the group that the CNN model tends to overfit early which is also noticeable in the base study. Hence, another recommendation is to use another model to be paired with BERT such as LSTM which is a more capable model in terms of natural language processing. At the same time, another option is to further experiment with the hyperparameters of CNN, especially the dropout and weight decay of the Adam optimizer which is set to 0.5 and 0.0 respectively which is similar to the base model. Alternatively, researchers can also modify the architecture of the CNN model as the architecture may be too complicated which led to early overfitting.
2. Additional Data for Model Training and Testing: The researchers recommend incorporating additional data to enrich the general context input for BERT and CNN algorithms, aiming to broaden the understanding of language patterns, sentiments, and contexts associated with hate speech. This approach, grounded in the idea of utilizing a diverse and extensive dataset, is expected to enhance model

generalization across various scenarios and improve predictive capabilities. The inclusion of more data not only enhances model robustness but also enables effective performance on diverse inputs, particularly in dynamic and context-dependent language applications, such as those found in online social platforms.

3. BERT Embeddings as the Input of CNN: The researchers suggest exploring the use of BERT embeddings as the input for CNN models. BERT embeddings capture rich semantic information about words and their contextual relationships, and integrate them as input features for a CNN. Combining the strengths of both models may lead to a more comprehensive representation of text data, potentially improving the overall performance in hate speech detection tasks. This approach allows for flexibility and innovation in the choice of model architectures, fostering advancements in the field of natural language processing and hate speech detection.
4. Further Experiments with Hashtags: At the same time, the researchers also recommended experimenting with hashtags like removing the hashtag symbol "#" instead of removing the hashtags entirely as this study showed that BERT is capable of handling them but its embedded value is not as significant as the researchers hoped. However, the results of this might end up the same as the candidate names as it might tend to misclassify inputs because of heavy embeddings.
5. Extension to Other Domains: The extension of this study to evaluate the generalizability of the BERT-CNN model across diverse domains and contexts is

vital. Assessing its performance outside the realm of Philippine election-related posts provides valuable insights into the model's versatility and adaptability. Such an extension could contribute to the development of more universally applicable hate speech detection systems.

By embracing these recommendations, future research endeavors can build upon the foundation laid by this study and advance the field of hate speech detection towards more nuanced, contextually aware, and adaptable models.

References

Alfina, I., Sigmawaty, D., Nurhidayati, F., & Hidayanto, A. N. (2017). Utilizing hashtags for sentiment analysis of tweets in the political domain. *Proceedings of the 9th International Conference on Machine Learning and Computing*. https://doi.org/10.1145/3055635.3056631

Alzahrani, E., & Jololian, L. (2021). How different text-preprocessing techniques using the BERT model affect the gender profiling of authors. *Advances in Machine Learning*. https://doi.org/10.5121/csit.2021.111501

Alim, M. M. F. (2021, June). A sentiment analysis study for Twitter using the various model of convolutional neural network. In *Journal of Physics: Conference Serie*s (Vol. 1918, No. 4, p. 042136). IOP Publishing.

Argañosa, S. C. D., Marasigan, R. L., Villanueva, J. J. S., & Wenceslao, J. K. C. (2022). Hate Speech in Filipino Election-Related Tweets: A Sentiment Analysis Using Convolutional Neural Networks.

Baluyut, M. K. A., dela Resma, M., Fernando, A. R. , Lapid, N., Sy, S. R., Timajo A. (2022). Where Is the Love? Identifying Hate Speech in Philippine Election-Related Tweets. *Asian Institute of Management.*

Bello, A., Ng, S.-C., & Leung, M.-F. (2023). A BERT framework to sentiment analysis of Tweets. *Sensors*, 23(1), 506. https://doi.org/10.3390/s23010506

Boquiren, A. J. V., Garcia, R. A., Hungria, C. J. D., & de Goma, J. C. (2022). Tagalog Sentiment Analysis Using Deep Learning Approach With Backward Slang Inclusion.

Brownlee, J. (2018). Statistical Significance Tests for Comparing Machine Learning Algorithms. *Machine Learning Mastery [online]*, 20.

Chan, S., & Fyshe, A. (2018). Social and emotional correlates of capitalization on Twitter. ACL Anthology. https://aclanthology.org/W18-1102/

Chiorrini, A., Diamantini, C., Mircoli, A., & Potena, D. (2021). Emotion and sentiment analysis of tweets using BERT. In *EDBT/ICDT Workshops* (Vol. 3).

Dao, T. A., & Aizawa, A. (2023, June). Evaluating the Effect of Letter Case on Named Entity Recognition Performance. In *International Conference on Applications of Natural Language to Information Systems* (pp. 588-598). Cham: Springer Nature Switzerland.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dong, J., He, F., Guo, Y., & Zhang, H. (2020, May). A commodity review sentiment analysis based on BERT-CNN model. In *2020 5th International conference on computer and communication systems (ICCCS)* (pp. 143-147). IEEE.

Eisenstein, J. (2019). *Introduction to natural language processing*. MIT press.

Exxact. (2021, March 29). Bert Transformers – how do they work? https://www.exxactcorp.com/blog/Deep-Learning/how-do-bert-transformers-work

Galinato, V., Amores, L., Magsino, G. B., & Sumawang, D. R. (2023). Context-Based Profanity Detection and Censorship Using Bidirectional Encoder Representations from Transformers. *Available at SSRN 4341604.*

Gambäck, B., & Sikdar, U. K. (2017). Using Convolutional Neural Networks to Classify Hate-Speech. *Proceedings of the First Workshop on Abusive Language Online.* https://doi.org/10.18653/v1/w17-3013

Hapke, H., Howard, C., & Lane, H. (2019). *Natural Language Processing in Action: Understanding, analyzing, and generating text with Python.* Simon and Schuster.

Hate speech and incitement to hatred or violence. OHCHR. (n.d.). https://www.ohchr.org/en/special-procedures/sr-religion-or-belief/hate-speech-and-incitement-hatred-or-violence#:~:text=As%20a%20matter%20of%20principle,peaceful%2C%20inclusive%20and%20just%20societies.

Hidayatullah, A. F., Cahyaningtyas, S., & Hakim, A. M. (2021). Sentiment analysis on Twitter using neural network: Indonesian presidential election 2019 dataset. *IOP Conference Series: Materials Science and Engineering, 1077(1), 012001.* https://doi.org/10.1088/1757-899x/1077/1/012001

Imperial, J. M., Orosco, J., Mazo, S. M., & Maceda, L. (2019). Sentiment Analysis of Typhoon Related Tweets using Standard and Bidirectional Recurrent Neural Networks. *arXiv preprint arXiv:1908.01765.*

Kaur, K., & Kaur, P. (2023). Improving BERT model for requirements classification by bidirectional LSTM-CNN deep model. *Computers and Electrical Engineering, 108,* 108699.

Kaviani, M., & Rahmani, H. (2020). EmHash: Hashtag Recommendation using Neural Network based on BERT Embedding. Sci-hub. https://sci-hub.se/10.1109/icwr49608.2020.9122296

Leung, K. (2022). Micro, Macro & Weighted Averages of F1 Score, Clearly Explained. Diambil kembali dari Towards Data Science: https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f.

Li, W., Gao, S., Zhou, H., Huang, Z., Zhang, K., & Li, W. (2019, December). The automatic text classification method based on BERT and feature union. In *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)* (pp. 774-777). IEEE.

Liu, B. (2020). Sentiment analysis: *Mining opinions, sentiments, and emotions*. Cambridge university press.

Lutkevich, B. (2020). What is BERT (Language Model) and How Does It Work?.

Malik, P., Aggrawal, A., & Vishwakarma, D. K. (2021, April). Toxic speech detection using traditional machine learning models and BERT and *fastText* embedding with deep neural networks. In *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)* (pp. 1254-1259). IEEE.

Mehta, R. P., Sanghvi, M. A., Shah, D. K., & Singh, A. (2019). Sentiment analysis of tweets using supervised learning algorithms. *First International Conference on Sustainable Technologies for Computational Intelligence*, 323–338. https://doi.org/10.1007/978-981-15-0029-9_26

Mingua, J., Padilla, D., & Celino, E. J. (2021, November). Classification of Fire Related Posts on Twitter Using Bidirectional Encoder Representations from Transformers (BERT). In *2021 IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)* (pp. 1-6). IEEE.

Nanli, Z., Ping, Z., Weiguo, L. I., & Meng, C. (2012, November). Sentiment analysis: A literature review. In *2012 International Symposium on Management of Technology (ISMOT)* (pp. 572-576). IEEE.

Regalado, R. V. J., & Cheng, C. K. (2012, November). Feature-based Subjectivity Classification of Filipino Text. In *2012 International Conference on Asian Language Processing* (pp. 57-60). IEEE.

Roy, A., & Ojha, M. (2020). Twitter sentiment analysis using Deep Learning Models. *2020 IEEE 17th India Council International Conference (INDICON).* https://doi.org/10.1109/indicon49873.2020.9342279

Safaya, A., Abdullatif, M., & Yuret, D. (2020, December). Kuisail at semeval-2020 task 12: BERT-CNN for offensive speech identification in social media. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation* (pp. 2054-2059).

Shaikh, R. (2023, August 29). Mastering Bert: A comprehensive guide from beginner to advanced in natural language processing... Medium. https://medium.com/@shaikhrayyan123/a-comprehensive-guide-to-understanding-bert-from-beginners-to-advanced-2379699e2b51

Solitana, N. T., & Cheng, C. K. (2021, December). Analyses of Hate and Non-Hate Expressions during Election using NLP. In *2021 International Conference on Asian Language Processing (IALP)* (pp. 385-390). IEEE.

Sunarya, P. A., Refianti, R., Mutiara, A. B., & Octaviani, W. (2019). Comparison of accuracy between convolutional neural networks and Naïve Bayes Classifiers in sentiment analysis on Twitter. *International Journal of Advanced Computer Science and Applications*, 10(5).

Tariq, A. (n.d.). What is the difference between micro and macro averaging?. Educative: https://www.educative.io/answers/what-is-the-difference-between-micro-and-macro-averaging

Velankar, A., Patil, H., Gore, A., Salunke, S., & Joshi, R. (2022). L3cube-mahahate: A tweet-based marathi hate speech detection dataset and BERT models. *arXiv preprint arXiv:2203.13778*.

Appendix A

Snippet of Dataset for Multi Label Classification

Tweet Content	Sentiment	Label
Dictator's son on the cusp of power in the #Philippines #marcos https://t.co/5wbFYVWlG4	Negative	Hate
Kung ang definition ni BongBong Marcos sa isang terorista ay isang armadong grupo na nananakit, nanunupil ng civilian o teenager na NON COMBATANT, pinapatunayan lang na ang PNP-AFP ay terorista. EJK sa Drug War at Martial Law mismo ang ebidensya. #ManggagawaVSMagnanakaw	Negative	Hate
Leni Robredo is not being subjected by her opponents to gender attact but her intelligence and capacity to lead are put into question... we do not want a puppet president... https://t.co/hzn3LiQ5uJ	Negative	Hate
@imstillsour Tuwang tuwa sila sa Pink hahaha..hindi nila alam Pink color is sign for Bongbong Marcos..kasi favorite nya isuot pink...ambilis ni Leni mag grab ng color kasi nakita nila langit nag Pink hahahaðŸ˜…ðŸ˜‚ðŸ˜‚,	Negative	Hate
Kapag hindi si VP Leni Robredo ang iboboto mo sa May 9 2022!! 6 YEARS KANG MAMALASIN...	Negative	Hate
kakampinks sleeping soundly tonight knowing that leni robredo did well in tonightâ€™s debate. pero yung iba, double time sa paghahanap ng mali. ðŸ˜‚, hirap talaga pag wala ka mapagmalaki sa kandidato mo sa iba ka nalang titingin. #10RobredoPresident #IpanaloNa10To	Negative	Hate
leni robredo daw potanginaAAAAUSASASSASA	Negative	Hate
Son of a Dictator | The rise of FERDINAND MARCOS JNR - This week on FOREIGN CORRESPONDENT Read More -> https://t.co/FNivaeF42j https://t.co/HWGtOLR2e5	Negative	Hate
Fascinating read ðŸ‘‡ðŸ¼		

Appendix B

Snippet of Dataset for Binary Label Classification

text	sentiment	label
Bongbong Marcos absent pa rin sa Comelec-sponsored presidential debate https://t.co/7xvtEJECGu	Negative	Hate
In the Philippines, presidential frontrunner Bongbong Marcos is using TikTok and troll farms to rewrite his dictator father's violent past for young Filipinos (David Pierson/Los AngelesÂ Times) https://t.co/f53nhpL5LV	Negative	Hate
THE DELUDED ONE Leni Robredo, a Filipino politician who claims to be a mother of a nation but majority of Filipinos refuse to accept. https://t.co/fjwZSXip9c	Negative	Hate
@ANCALERTS @_katrinadomingo Walang kaaway at hindi nang-aaway si Leni Robredo. Matino at matapat na serbisyo sa Pilipino ang plataporma nya. Matagal na ngang patay si marcos pero di mabubura ang katotohanan na diktador at magnanakaw sya! Maraming nagdusa sa kagagawan nya!	Negative	Hate
â€˜The young ones, they donâ€™t knowâ€™: How a dictatorâ€™s dark past was whitewashed https://t.co/LI0eP8VR3e #news #berkleybearnews	Negative	Hate
Pag may Bongbong Marcos supporter na nagsabi sa iyong		

Appendix C

Sample List of Negative Sentiment Hashtags from Multi Label Classification Dataset

text	sentiment	label
#Philippines #marcos	Negative	Hate
#ManggagawaVSMagnanakaw	Negative	Hate
#10RobredoPresident #IpanaloNa10To	Negative	Hate
#NagaIsPink #BicolIsPink	Negative	Hate
#PiliPinas2022 #Eleksyon2022	Negative	Hate
#NagaIsPink #JusticeForFrederick	Negative	Hate
#NagaIsPink #BicolIsPink	Negative	Hate
#LeniDuwag	Negative	Hate
#kakampINC #LabanLeni2022 #LabanLeniKiko2022	Negative	Hate
#BBMSARACONFERENCE	Negative	Hate
#NeverAgaintoLiberals	Negative	Hate
#loren	Negative	Hate
#ashsgs	Negative	Hate
#CrimePays #Dictators #Putin #Russia #Ukraine	Negative	Hate
#LetLeniKikoLead	Negative	Hate
#Ipana7oNa10to #LetLeniKikoLead	Negative	Hate
#PiliPinasDebates2022	Negative	Hate
#LeniKikoAllTheWay2022	Negative	Hate
#LeniKiko2022	Negative	Hate
#IskoMoreno #BilisKilos	Negative	Hate
#IglesiaNiCristo #KakampINC #LenSlide	Negative	Hate
#PiliPinasDebates2022	Negative	Hate
#PiliPinasDebates2022	Negative	Hate
#B	Negative	Hate
#UPinkFight #NagaIsPink	Negative	Hate
#PiliPinasDebates2022	Negative	Hate
#Halalan2022	Negative	Hate
#CNNPHVPDebate	Negative	Hate
#NagaIsPink	Negative	Hate
#LoserSign #LeniLoser	Negative	Hate
#GoBBMforSMNI #SMNIpresidentialdebate #LENIDUWAG	Negative	Hate
#ASIAPACIFIC	Negative	Hate
#budol #PingLacson #NeverForget #MarcosDuwag	Negative	Hate
#LetLeniLeave2022 #BBMIsMyPresident2022	Negative	Hate
#Elections2022 #SalamatPRRD	Negative	Hate
#PiliPinasDebates2022	Negative	Hate
#IpanaloNa10To	Negative	Hate
#news #topstories #berkleybearnews	Negative	Hate
#IskoWithdraw #LeniforPresident2022	Negative	Hate
#SMNIpresidentialdebate	Negative	Hate
#BBMIsMyPresident2022	Negative	Hate
#LeniRobredo2022 #LabanLeni2022	Negative	Hate
#MarcosMagnanakaw #MarcosDuwagParin	Negative	Hate
#BBMSARA2022 #UniTeamUniteInILOILO	Negative	Hate
#BBMIsMyPresident2022	Negative	Hate
#nft #tezos	Negative	Hate

Appendix D

Sample List of Positive Sentiment Hashtags from Multi Label Classification Dataset

#KayPingSigurado #HindiSayangAngBotoMo #PiliPinasDet	Positive	Non-hate
#TaraNaKayLeni #10RobredoForPresident #PiliPinasDeba	Positive	Non-hate
#GobyernongTapatAngatBuhayLahat #LeniRobredo	Positive	Non-hate
#Kakamping	Positive	Non-hate
#CoRRECTtheConstitution	Positive	Non-hate
#IpanaloNa10To #10RobredoPresident #CNNPHPresident	Positive	Non-hate
#ReginеVelasquez #MaricelSoriano #OgieAlcasid #Sharon	Positive	Non-hate
#Twibbon	Positive	Non-hate
#MBBhindiBBM #LeniKikoAllTheWay2022 #KulayRosasAn	Positive	Non-hate
#IpanloNa10ToParaSaLahat #AngatBuhayLahat #LeniKiko,	Positive	Non-hate
#LetLeniLead2022 #10RobredoForPresident #bbmanonga	Positive	Non-hate
#KulayRosasAngBukas #NewProfilePic	Positive	Non-hate
#Ipana10NatinParaSaLahat #LeniKiko2022 #LetLeniKikoLe	Positive	Non-hate
#PiliPinasDebates2022 #KayLeniNaTayo #KayLeniTayo #1	Positive	Non-hate
#PiliPinasDebates2022	Positive	Non-hate
#MakatiIsPink #MitingDeAvance #KulayRosasAngBukas #I	Positive	Non-hate
#IpanloNa10ToParaSaLahat #LetLeniKikoLead	Positive	Non-hate
#ViceGanda #MakatiIsPink #PilipinasIsPink #Ipana7oNa10	Positive	Non-hate
#PingLacsonHanggangDulo #5PresidentePingLacson #Lacs	Positive	Non-hate
#PiliPinasDebates2022	Positive	Non-hate
#KulayRosasAngBukas #LetLeniKikoLead2022	Positive	Non-hate
#Ipana7oNa10ParaSaLahat	Positive	Non-hate
#BringBackMarcos #MarcosDuterte2022 #PROTECTBBMS	Positive	Non-hate
#KayLeniNaTayo	Positive	Non-hate
#10RobredoPresident #LeniAngatSaLahat #CNNPHPresid	Positive	Non-hate
#LetLeniLead #KulayRosasAngBukas #GobyernongTapatA	Positive	Non-hate
#SinongPangGiyeraMo #PingLacsonTayo #PiliPinasDebate	Positive	Non-hate
#ILOILOisPink #GrandPasasalamat #IpanloNa10ParaSaLah	Positive	Non-hate
#10RobredoForPresident #LeniKikoAllTheWay2022	Positive	Non-hate
#LeniKiko2022	Positive	Non-hate
#IpanaloNa10To	Positive	Non-hate
#TaraNaKayLeni	Positive	Non-hate
#IpanaloNa10ParaSaLahat #LeniKiko2022	Positive	Non-hate
#LeniKikoAllTheWay #LeniKikoAllTheWay2022 #DonBelle	Positive	Non-hate
#FaceTheSun_D21 #ì„¸ë¸´ #SEVENTEEN	Positive	Non-hate
#KulayRosasAngBukas #AngatBuhayLahat #kakampinkwe	Positive	Non-hate
#LetLeniKikoLead #KulayRosasAngBukas #LeniKiko2022	Positive	Non-hate
#Ipana7oNa10to #PilipinasIsPink #Ipana10NatinParaSaLal	Positive	Non-hate
#Twibbon	Positive	Non-hate
#KayLeniNaTayo #LetLeniLead	Positive	Non-hate
#DapatSiLeni #10RobredoPresident #IpanaloNa10to #Len	Positive	Non-hate
#KayLeniNaTayo #10RobredoPresident #PiliPinasDebates	Positive	Non-hate
#CNNPresidentialDebate #KulayRosasAngBukas #Kakamp	Positive	Non-hate
#LeniKikoAllTheWay2022 #ACEsForLeniKiko	Positive	Non-hate
#KulayRosasAngBukas #LeniKikoAllTheWay2022	Positive	Non-hate
#KakampINC	Positive	Non-hate
#KulayRosasAngBukas #Kakampink #LeniKiko2022	Positive	Non-hate

Appendix E

Sample List of Neutral Sentiment Hashtags from Multi Label Classification Dataset

#10RobredoForPresident #AngatBuhayLahat	Neutral	Non-hate
#LeniTunayNaISKO	Neutral	Non-hate
#SMNIpresidentialdebate #GoBBMforSMNI #BBMForPres	Neutral	Non-hate
#OneMindanao #GMARegionalTV #Eleksyon2022	Neutral	Non-hate
#205	Neutral	Non-hate
#PiliPinasDebates2022 #SinongPangGiyeraMo #LacsonSo	Neutral	Non-hate
#PilipinasDebates2022 #PTVElectionTV	Neutral	Non-hate
#NagaIsPink #BicolIsPink	Neutral	Non-hate
#SMNIpresidentialdebate	Neutral	Non-hate
#SMNIpresidentialdebate #BBMForPresident #LENIDUWA	Neutral	Non-hate
#Elections2022PH	Neutral	Non-hate
#KulayRosasAngBukas #IpanaloNa10ParaSaLahat #IbotoN	Neutral	Non-hate
#LetLeniKikoLead #LeniKiko2022	Neutral	Non-hate
#KayLeniNaTayo	Neutral	Non-hate
#ManggagawaVSMagnanakaw	Neutral	Non-hate
#TheChronicle #YourNewsNOW #SinoBayan	Neutral	Non-hate
#KayLeniNaTayo #KayLeniTayo #PiliPinasDebates2022	Neutral	Non-hate
#LabanLeni2022	Neutral	Non-hate
#KakamPink	Neutral	Non-hate
#KayLeniNaTayo	Neutral	Non-hate
#TaraNaKayLeni #PiliPinasDebates2022	Neutral	Non-hate
#BilangPilipino2022	Neutral	Non-hate
#BilangPilipino2022	Neutral	Non-hate
#BBMIsMyPresident2022 #BBMSaraUNITEAM #HALALAN	Neutral	Non-hate
#TaraNaKayLeni #PiliPinasDebates2022	Neutral	Non-hate
#NoToMarcosDuterte2022	Neutral	Non-hate
#SinongPangGiyeraMo #PingLacsonTayo #PiliPinasDebate	Neutral	Non-hate
#LetLeniLead2022	Neutral	Non-hate
#PHElections2022 #OurTurn2022	Neutral	Non-hate
#TaraNaKayLeni #PilipinasDebates2022	Neutral	Non-hate
#PiliPinasDebates2022	Neutral	Non-hate
#LetLeniLead #TeamLeniRobredo #LabanLeni2022 #Dapa	Neutral	Non-hate
#PiliPinasDebates2022 #SwitchToIsko #IskoMoreno	Neutral	Non-hate
#PiliPinasDebates2022 #TaraNaKayLeni #LeniRobredoFor	Neutral	Non-hate
#LetLeniKikoLead #Ipana7oNa10to #CebuYass	Neutral	Non-hate
#SwitchToIsko	Neutral	Non-hate
#AbogadoSnitch	Neutral	Non-hate
#TaraNaKayLeni #PiliPinasDebates2022	Neutral	Non-hate
#PHVote #WeDecide	Neutral	Non-hate
#SinoPangGiyeraMo #LacsonSottoPanaloTayo #PINGdigm	Neutral	Non-hate
#LeniKiko2022 #KNsaHalalan2022	Neutral	Non-hate
#BotoKo2022	Neutral	Non-hate
#BBMduwag	Neutral	Non-hate
#NorbertoGonzales #SocialDemocracyPh #FirstWorldPhili	Neutral	Non-hate
#LeniKikoAllTheWay2022 #LeniKiko2022 #LeniForPreside	Neutral	Non-hate
#SaveLegendsOfTomorrow	Neutral	Non-hate

Appendix F

Sample List of Hate Labels Hashtags from Binary Label Classification Dataset

text	sentiment	label
#news #berkleybearnews	Negative	Hate
#BBMSARA2022 #INC	Negative	Hate
#Ipana7oNa10to	Negative	Hate
#PiliPinasDebates2022 #TheFilipinoVotes	Negative	Hate
#PiliPinasDebates2022	Negative	Hate
#PilipinasIsPink #MakatiIsPink	Negative	Hate
#COCfiling #PHVote #WeDecide #InTheRunning	Negative	Hate
#markanthonyvillanueva #FredrickAlba #LeniRobredo	Negative	Hate
#Ipana7oNa10ParaSaLahat #Ipana10NatinParaSaLahat	Negative	Hate
#MarcosSinungaling	Negative	Hate
#GoBBMforSMNI #SMNIpresidentialdebate #LENIDUWAG #LeniMasDuwag #LeniTangaSaLahat	Negative	Hate
#ChoosePing #5PresidentePingLacson #LacsonMostQualifiedPresident #PiliPinasDebates2022	Negative	Hate
#bardagulan	Negative	Hate
#SinongPangGiyeraMo #PingLacsonTayo #PiliPinasDebates2022	Negative	Hate
#UPinkFight #NagaIsPink	Negative	Hate
#BBMLoser	Negative	Hate
#LetLeniLead2022 #Kakampinks	Negative	Hate
#PilipinasIsPink #KulayRosasAngBukas #KayLeniPatayTayo #LeniKikoMitingDeAvance	Negative	Hate
#SMNIpresidentialdebate	Negative	Hate
#LetLeniLead2022 #LeniforPresident2022	Negative	Hate
#GoBBMforSMNI #SMNIpresidentialdebate #LENIDUWAG #LeniMasDuwag #LeniTangaSaLahat	Negative	Hate
#PiliPinasDebates2022	Negative	Hate
#GoBBMforSMNI #SMNIpresidentialdebate #LENIDUWAG #LeniMasDuwag #LeniTangaSaLahat	Negative	Hate
#PiliPinasDebates2022 #TheFilipinoVotes	Negative	Hate
#NagaIsPink #MitingDeAvance #UPinkFight #BicolIsPink #IbotoNa10Pilipinas	Negative	Hate
#ashsgs	Negative	Hate
#KpopstansinREDandGREEN	Negative	Hate
#ManggagawaVSMagnanakaw	Negative	Hate
#unitedalamano	Negative	Hate
#TaraNaKayLeni	Negative	Hate
#halalanspace	Negative	Hate
#CNNPHPresidentialDebate	Negative	Hate
#LeniLugaw #LeniLutang	Negative	Hate
#LeniRobredoForPresident #PiliPinasDebates2022	Negative	Hate
#NOTOLENI	Negative	Hate
#NeverAgain	Negative	Hate
#ashsgs	Negative	Hate
#KulayRosasAngBukas #KayLeniPatayTayo #LeniKikoMitingDeAvance	Negative	Hate
#ashsgs	Negative	Hate
#PiliPinasDebates2022	Negative	Hate

Appendix G

Sample List of Non-Hate Labels Hashtags from Binary Label Classification Dataset

#IbotoNa10Pilipinas #IpanaloNa10ParaSaLahat	Positive	Non-hate
#From #to #KakampINC #LetLeniLead #LeniRobredo #LeniKiko #LenSlide	Positive	Non-hate
#PHVote #WeDecide	Positive	Non-hate
#KayLeniNaTayo	Positive	Non-hate
#LeniKikoAllTheWay2022 #ACEsForLeniKiko	Positive	Non-hate
#IpanaloNa10To	Positive	Non-hate
#PINGdigmaan #LacsonSottoPanaloTayo #kakamPing #lacson	Positive	Non-hate
#AdamsonianParaKayLeniKiko #LeniKiko2022	Positive	Non-hate
#LetLeniKikoLead #LeniKiko2022 #KulayRosasAngBukas #SaGobyernongTapatAngatBuhayLahat	Positive	Non-hate
#LetLeniKikoLead #LeniKiko2022 #KulayRosasAngBukas #SaGobyernongTapatAngatBuhayLahat #LenSlide #SiKikoAngManokKo	Positive	Non-hate
#PBBTeen5thEviction	Positive	Non-hate
#TaraNaKayLeni #PiliPinasDebates2022	Positive	Non-hate
#LetLeniLead #KulayRosasAngBukas	Positive	Non-hate
#VPLeniRobredo #MyPresident #KayLeniNaTayo #10RobredoPresident #kayleniangatbuhaylahat	Positive	Non-hate
#Ping	Positive	Non-hate
#LENI	Positive	Non-hate
#KulayRosasAngBukas #AngatBuhayLahat #LeniKikoAllTheWay	Positive	Non-hate
#LetLeniKikoLead #LeniKiko2022 #KulayRosasAngBukas #NowUnited	Positive	Non-hate
#5PresidentePingLacson #LacsonMostQualifiedPresident #PiliPinasDebates2022	Positive	Non-hate
#leni #robredo	Positive	Non-hate
#TaraNaKayLeni #PiliPinasDebates2022 #10RobredoForPresident	Positive	Non-hate
#Ipana7oNa10to #IpanaloParaSaLahat #LetLeniKikoLead #LeniKiko2022 #KulayRosasAngBukas #SaGobyernongTapatAngatBuhayLahat #LenSlide #SiKikoAngManokKo	Positive	Non-hate
#SMNI #SMNIpresidentialdebate	Positive	Non-hate
#PiliPinasDebates2022	Positive	Non-hate
#LeniKiko2022 #LeniKikoAllTheWay	Positive	Non-hate
#Ipana7oNa10To #LeniKiko2022	Positive	Non-hate
#10 #CNNPresidentialDebate #IpanaloNa10To #10RobredoForPresident	Positive	Non-hate
#PiliPinasDebates2022 #KayLeniNaTayo #KayLeniTayo #10RobredoForPresident	Positive	Non-hate
#BBMSaraUNITEAM #7 #4 #BBMFORPRESIDENT2022 #BBMIsMyPresident2022	Positive	Non-hate
#KakampINC #LeniKikoAllTheWay	Positive	Non-hate
#LabanLeni2022 #LetLeniLead2022 #DapatSiLeni	Positive	Non-hate
#PingLacson2022	Positive	Non-hate
#AngatBuhayPilipino #LeniKikoAllTheWay2022 #10RobredoForPresident #07PangilinanForVicePresident	Positive	Non-hate
#NadineLustre #NeverAgain #SiargaoIsPink #KayLeniTayo #IpanaloNa10To #PampangaIsPink	Positive	Non-hate
#KayLeniNaTayo	Positive	Non-hate

Appendix H

Sample List of Negative Sentiment Text with ALL-CAPS from Multi Label Classification Dataset

text	sentiment	label
dictator's son on the cusp of power in the	Negative	Hate
kung ang definition ni bongbong marcos sa isang terorista ay isang armadong grupo na nananakit nanunupil ng civilian teenager na NON COMBATANT pinapatunayan lang na ang PNP-AFP ay terorista EJK sa drug war at martial law mismo ang ebidensya	Negative	Hate
leni robredo is not being subjected by her opponents to gender attact but her intelligence and capacity to lead are put into question we do not want puppet president	Negative	Hate
tuwang tuwa sila sa pink hahaha..hindi nila alam pink color is sign for bongbong marcos..kasi favorite nya isuot pink...ambilis ni leni mag grab ng color kasi nakita nila langit nag pink	Negative	Hate
kapag hindi si VP leni robredo ang iboboto mo sa may YEARS KANG MAMALASIN	Negative	Hate
kakampinks sleeping soundly tonight knowing that leni robredo did well in tonight's debate pero yung iba double time sa paghahanap ng mali hirap talaga pag wala ka mapagmalaki sa kandidato mo sa iba ka nalang titingin	Negative	Hate
leni robredo daw potanginaAAAAUSASASSASA	Negative	Hate
son of dictator the rise of FERDINAND MARCOS JNR this week on FOREIGN CORRESPONDENT read more gt	Negative	Hate
fascinating read why bongbong marcos philippine dictator's son leads the race for the presidency	Negative	Hate
lol walang kwenta yang pagmamagaling nyo ano ang ginawa nyo for years di ba bullying din ngayon sasabihin nyo tinuturuan ni joma sison si leni robredo	Negative	Hate
boy ngiwi didn't even finish his undergraduate studies at oxford in sum nasipa sya tapos yun tatay dahil pangulo humingi ng some sort of fake diploma wala syang diegree so he lies about ut oxford amp wharton DO NOT HAVE LBM ON THEIR LIST OF GRADS	Negative	Hate
president leni robredo thats an article in new yor times,,nakakahiya ang pilipinas boboto sa anak ng diktador ipanalo leni robredo	Negative	Hate
bongbong marcos the man attempting to revive corrupt political dynasty his father was ruthless dictator his mother gained international notoriety for her massive collection of shoes	Negative	Hate
in the philippines grass-roots campaign takes on the marcos juggernaut ugh son of dictator marcos is running for president hoping VP maria leonor leni robredo amp her digital warriors truth army can defeat the marcos duterte paid propagandists	Negative	Hate
mamalasin tayo dahil inuna pang pangarapin ni cj cansino makatuntong sa malacang with leni robredo nauna na syang minalas bangko tuloy	Negative	Hate
ang totoong sayang na boto ay para sa mga hindi karapat-dapat manalo at sa mga magnanakaw na dapat walang karapatang maluklok sa puwesto yan ang closing message ni ping lacson sa ikalawang comelec presidential debate	Negative	Hate

Appendix I

Sample List of Positive Sentiment Text with ALL-CAPS from Multi Label Classification Dataset

nakita nyo naman kung gaano ka tuwid at kungkreto ang sagot ni senator ping lacson unang tanong palang yan mag-ingaym ga	Positive	Non-hate
bakit si leni robredo	Positive	Non-hate
thank you sir norberto gonzales	Positive	Non-hate
tara na at ipakita ang ating buong pagsuporta sa ating mahusay at mabuting pinuno na si leni robredo sama-sama nating ipanalo to	Positive	Non-hate
the best man for the job is woman leni robredo this is so powerful	Positive	Non-hate
sa gobyernong tapat angat buhay lahat ako si hariette isang iskolar ng bayan nakiki-isa sa pagsuporta kay vice president leni robredo at senator kiko pangilinan kasama ang tropa ng senatorial slate para sa bayan na matagal na nating pinapangarap	Positive	Non-hate
lol kami ay para kay senator ping lacson	Positive	Non-hate
president leni robredo cutie	Positive	Non-hate
last night beer place somewhere in maginhawa all of sudden table started chanting ang presidente leni robredo bise presidente kiko pangilinan and two more tables joined in did too it was electric if surveys cant detect this they were programmed not to	Positive	Non-hate
what has gotten into this norberto gonzales that he continues to allude that kakampinks are prepping for people power if VP leni loses what's his proof nakaka inis na sila ha	Positive	Non-hate
iba talaga mama mo tagos sa puso na naman ang ending statement our th president of the republic of the philippines isang ilaw ng tahanan..leni robredo!!may naghiwa ng sibuyas na naman	Positive	Non-hate
god bless sa lahat ng kakampink at higit sa lahat sa ating vp leni robredo lord ibalato nyo na samen si vp leni panalangin ng sambayanang pilipino na siya ang maging aming susunod na presidente para maibalik ang dangal at makatao ng lahat amen	Positive	Non-hate
president leni robredo cutie vice-president kiko pangilinan cutie	Positive	Non-hate
ang presidente LENI ROBREDO	Positive	Non-hate
sa huling pagkakataon ANG PRESIDENTE LENI ROBREDO BISE PRESIDENTE KIKO PANGILINAN you have my confidence and my vote	Positive	Non-hate
siguro kaya kalmado lang si VP leni kahit palapit na ang may kasi buong tao siya bago siya pumasok sa eleksyon na to hindi yung posisyon ang mag-dedefine ng pagkatao niya manalo matalo siya siya pa rin si leni robredo	Positive	Non-hate
changed my header consisting of lovely person and pink background days ago days later VP leni robredo announced her campaign for presidency and her party color is pink who knew that jung ho-yeon and VP leni's announcement would fill my heart with love amp joy	Positive	Non-hate
because VP leni had enough and chose violence for today's videyow as usual VP leni robredo did not disappoint facts based and data based answers and tomorrow let's watch and show our support to sen	Positive	Non-hate
president leni robredo on may cutie	Positive	Non-hate
will pray that leni robredo will be the president of the philippines	Positive	Non-hate
ANG PRESIDENTE LENI ROBREDO BISE PRESIDENTE KIKO PANGILINAN	Positive	Non-hate

Appendix J

Sample List of Neutral Sentiment Text with ALL-CAPS from Multi Label Classification Dataset

ay fake news pala yung pupunta si leni robredo sa vigan this saturday sayang umasa pa naman ako ng bongga	Neutral	Non-hate
faisal mangondato baga	Neutral	Non-hate
opinion imelda marcos's shoe collection was glimpse into frightening reign	Neutral	Non-hate
happy mothers day to the future mother of our country VP leni robredo pres leni robredo cutie	Neutral	Non-hate
SMNI debate ernie abella is actually not so bad wow	Neutral	Non-hate
SIGN NA RAW TO PARA IBOTO NIYO SI LENI ROBREDO AT CHEL DIOKNO	Neutral	Non-hate
LENI	Neutral	Non-hate
told myself won't change my display icon clouds and the sky unless we'll get chanbaek collab but added pink twb to show my support for leni robredo and kiko pangilinan	Neutral	Non-hate
leni robredo's closing statement in tonight's debate	Neutral	Non-hate
VP leni corruption system or the people how are you going to solve it robredo parehong may problema ways to solve it accountability transparency people empowerment	Neutral	Non-hate
sa ating pakikipag-#taosataokayrobredo at patuloy na pagkampanya para sa isang gobyernong tapat na mag-aangat sa lahat dumarami na ang ating mga kababayan na narito ang kwento nila mula sa VP leni robredo FB page	Neutral	Non-hate
with less than week before the elections former senator antonio trillanes IV is giving four tips on how to propel the chances of vice president leni robredo win the presidential elections	Neutral	Non-hate
basta love bongbong marcos color brown lang malakas	Neutral	Non-hate
the reason why we are still importing rice is because of the lack of support for the agriculture sector it all boils down to our governance that's why please don't leave out kiko for every leni robredo that you'll shade	Neutral	Non-hate
ipinagkibit-balikat nina presidential bets sen ping lacson at mayor isko moreno ang resulta ng huling survey ng pulse asia research inc naniniwala naman si sen manny pacquiao na maipapanalo siya ng pacman magic via	Neutral	Non-hate
president leni robredo	Neutral	Non-hate
PRESIDENT LENI ROBREDO ON MAY	Neutral	Non-hate
president robredo leni vice president pangilinan kiko	Neutral	Non-hate
sa FB lang po ito ah sa page po ni V)P leni hindi kasama ang live views ng news outlets ng mga volunteer pages i.e team leni robredo robredo people's council tropang angat chapters the like or even ang mga youtube channels ng mga nabanggit	Neutral	Non-hate
maria leonor leni gerona robredo th president of the republic of the philippines	Neutral	Non-hate
hinding-hindi ko pagsisihan na ang unang presidente at bise presidenteng iboboto ko ay si leni robredo at kiko pangilinan manalo matalo mananalo hindi ko ikakahiya na sumuporta ako sa tama	Neutral	Non-hate

Appendix K

Sample List of Hate Labeled Text with ALL-CAPS from Binary Label Classification Dataset

text	sentiment	label
bongbong marcos absent pa rin sa comelec-sponsored presidential debate	Negative	Hate
in the philippines presidential frontrunner bongbong marcos is using tiktok and troll farms to rewrite his dictator father's violent past for young filipinos david pierson/los angeles times	Negative	Hate
THE DELUDED ONE leni robredo filipino politician who claims to be mother of nation but majority of filipinos refuse to accept	Negative	Hate
pag may bongbong marcos supporter na nagsabi sa iyong payapa ang marcos era isampal mo sa kanya itong artikulo ni wanna ver anak mismo ni fabian ver military general ni marcos	Negative	Hate
dictator's son rewrites history on tiktok in his bid to become the philippines next president	Negative	Hate
ulol yung mga nag sasabing I'm an INC and vote for leni robredo ulol sino niloko nyo kapwa nyo kakampink hahaha pag lilinaw lang po sa lahat na yung mga nag popost ng ganyan is hindi talaga INC THANK YOU AND GODBLESS	Negative	Hate
mga kapwa ko pilipino hindi sana kami ganito makipaglaban ngayon kung alam naming magiging maganda ang pamamahala ni marcos jr ngunit hinde dahil sa kanyang track record NAPAKA DUMI kaya kailangan natinf ipanalo si president leni robredo	Negative	Hate
anyway tANGINA MO BONGBONG MARCOS SINUNGALING	Negative	Hate
if you still don't get this,pareho na talaga kayo ni bongbong marcos na	Negative	Hate
pakisupalpal sa mga DDS nating friends na biktima ng fake quote cards	Negative	Hate
BAKIT KAMI NAGBABAYAD NG UTANG NG MARCOS ADMINISTRATION HANGGANG NGAYON ISKO MORENO DOMAGOSO move on niya mukha niya sana hindi makalimutan ng maynila ang anti-poor policies niya	Negative	Hate
hope at the very least everyone agrees that VP leni robredo is cool and smart and bongbong marcos is an uncool spoiled brat who had everything handed to him including his fake diploma	Negative	Hate
friendly reminder pag si faisal mangondato na ang magsalita skip niyo na lang char if gusto niyo lang nman na hindi masayang oras niyo charotz atleast siya nagpapakita HAHAHA	Negative	Hate
sa bus kanina they asked me kung sino ang presidente ko when answered ka leody de guzman po they were lyk sino yorn pinagtawanan pa nila ng mga conductor at driver na nagtanong sakin hindi ko sya madefend dahil hindi nila sya kilala	Negative	Hate
LOVE HOW HIGHLIGHTED THE ABSENCE OF DICTATOR'S SON FERDINAND BONGBONG MARCOS	Negative	Hate

Appendix L

Sample List of Non-Hate Labeled Text with ALL-CAPS from Binary Label Classification Dataset

PRESIDENT leni ROBREDO VICE PRESIDENT kiko PANGILINAN	Positive	Non-hate
gladly do NOT have fear for another marcos to lead the country I'm in one with the majority of the filipinos who are excited to have bongbong marcos as the th president of the PHILIPPINES propaganda na lang ang mga ganitong twitter space if not then desperate move	Positive	Non-hate
kahit iboto nyo na lang si leni robredo for president at kiko pangilinan as VP sa may as your birthday gift to me okay na okay na ako dun	Positive	Non-hate
bongbong marcos certificate of achievement booth award milo little olympics SOURCE	Positive	Non-hate
LENI ROBREDO NICKI MINAJ ERA kween	Positive	Non-hate
th president leni robredo cutie	Positive	Non-hate
QUEEN OF RESIBO SHOWING UP AT WOMANING UP MY PRESIDENT MARIA LEONOR LENI GERONA ROBREDO	Positive	Non-hate
president leni robredo did so great ops practice lang hahaha	Positive	Non-hate
yeah and that's president leni robredo	Positive	Non-hate
therefore can proudly say that my president is only she among the presidentiables vice pres leni robredo and soon to be the president leni	Positive	Non-hate
this is my first time voting so hindi ko sasayangin ang pagkakataon na maging parte ng pagbabago kung gusto natin umusad pumili dapat ng naayon sa pwesto wag natin sayangin ang chance na ito years din yan and so my vote goes for leni robredo	Positive	Non-hate
sa gobyernong tapat angat buhay lahat ako si faye isang iskolar ng bayan nakiki-isa sa pagsuporta kay vice president leni robredo at senator kiko pangilinan kasama ang tropa ng senatorial slate para sa bayan na matagal na nating pinapangarap	Positive	Non-hate
isko moreno ping lacson and ka leody did great job in that debate	Positive	Non-hate
true great leader has positive vision for himself and his people let's vote for senator ping lacson	Positive	Non-hate
ngayong darating na halalan ang tatanglaw sa bayan ay ilaw ng tahanan leni robredo my heart is soo full lord bigay mo na samin to please	Positive	Non-hate
puyat man lagi sa law school si kang sol mulat naman lagi sa katotohanan sa paninindigan at sa karapatang ipinaglalaban exercise your right to vote am ALL IN for VP leni robredo as my president and senator kiko pangilinan as my vice president	Positive	Non-hate
lt ang presidente leni robredo gt bise-presidente kiko pangilinan lt gt let's go	Positive	Non-hate

Appendix M

System Program in Jupyter Python

In [1]:
```
import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np
import torch.nn.functional as F
from transformers import AdamW
import pandas as pd
from transformers import BertTokenizer, BertModel, BertConfig
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
import csv
import re
import validators
import emoji
import unidecode
import nltk
import pickle
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to C:\Users\Mark Gabriel
[nltk_data]     Ortiz\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [2]:
```
# Set the seed for reproducibility
SEED = 1235
torch.manual_seed(SEED)
torch.backends.cudnn.deterministic = True

# BERT Hyperparameters (ADDITION)
n_bert_layers = 16  # Assuming the base model has 12 layers
bert_lr = 0.001
pooling_strategy = 'cls'  # Options: 'cls', 'mean', 'max'
bert_hidden_size = 1024  # Adjust based on your BERT model
max_seq_length = 128
fine_tune_strategy = 'last_layer'  # Options: 'full', 'last_layer'
bert_dropout = 0.9  # Adjust based on BERT model specifications

max_seq_length = 128  # This should match the max_seq_length used in BERT model
padding_strategy = 'max_length'  # Options: 'max_length', 'do_not_pad', 'longest'
truncation_strategy = 'longest_first'  # Options: 'longest_first', 'only_first', 'only_second'
do_lower_case = True  # Set to False if using a cased model

bert_model = BertModel.from_pretrained('bert-base-uncased')

# Obtain the configuration from the loaded model
config = bert_model.config

# Modify the configuration if necessary
n_bert_layers = 16  # Assuming you want to change the number of layers
bert_hidden_size = 1024  # Adjust based on your needs
bert_dropout = 0.9  # Adjust based on your needs

config.num_hidden_layers = n_bert_layers
config.hidden_size = bert_hidden_size
config.num_attention_heads = 16  # Adjust as needed
config.intermediate_size = 4 * bert_hidden_size  # Typically 4 times the hidden size
config.hidden_dropout_prob = bert_dropout
config.attention_probs_dropout_prob = bert_dropout

# Load BERT tokenizer and model
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased',
                                          max_length=max_seq_length,
                                          padding=padding_strategy,
                                          truncation=truncation_strategy,
                                          do_lower_case=do_lower_case)
# Load the BERT model with the custom configuration
bert_model = BertModel(config=config)
```

Binary:

In [3]:
```
data_path = 'binary-dataset.csv'
data_df = pd.read_csv(data_path)
data_df = data_df.rename(columns={'Tweet Content': 'text', 'Sentiment': 'sentiment', 'Label': 'label',})
```

In [4]:
```
data_df
```

Out[4]:

	text	sentiment	label
0	Bongbong Marcos absent pa rin sa Comelec-spons...	Negative	Hate
1	In the Philippines, presidential frontrunner B...	Negative	Hate
2	THE DELUDED ONE\n\nLeni Robredo, a Filipino po...	Negative	Hate
3	@ANCALERTS @_katrinadomingo Walang kaaway at ...	Negative	Hate
4	'The young ones, they don't know': How a dicta...	Negative	Hate
...	...	...	...
5115	LOOK: Dating Sen. Bongbong Marcos, nakikipagpu...	Neutral	Non-hate
5116	Gagi, nahihiwagaan talaga ako sa mga atabs na ...	Neutral	Non-hate
5117	manifesting circle: \n\n ...	Neutral	Non-hate
5118	"Ang tatanglaw sa ating bayan ay isang ILAW NG...	Neutral	Non-hate
5119	[2/2] On undocumented OFWs. PRESIDENT LENI ROB...	Neutral	Non-hate

5120 rows × 3 columns

Multi Label:

In [3]:
```
data_path = 'multilabel-dataset.csv'
data_df = pd.read_csv(data_path)
data_df = data_df.rename(columns={'Tweet Content': 'text', 'Sentiment': 'sentiment', 'Label': 'label'})
```

In [4]:
```
data_df
```

Out[4]:

	text	sentiment	label
0	Dictator's son on the cusp of power in the #Ph...	Negative	Hate
1	Kung ang definition ni BongBong Marcos sa isan...	Negative	Hate
2	Leni Robredo is not being subjected by her opp...	Negative	Hate
3	@imstillsour Tuwang tuwa sila sa Pink hahaha.....	Negative	Hate
4	Kapag hindi si VP Leni Robredo ang iboboto mo ...	Negative	Hate
...	...	...	...
7675	VP LENI ROBREDO FOR 2022 https://t.co/2ahSefgmST	Neutral	Non-hate
7676	@alt_ego143 @rapplerdotcom @ramboreports hindi...	Neutral	Non-hate
7677	I can't vote yet, but my president is Leni Rob...	Neutral	Non-hate
7678	Anyways President Leni Robredo #SaveLegendsOfT...	Neutral	Non-hate
7679	Kung ayaw nyo d wag nyo iboto.Napaka simple.. ...	Neutral	Non-hate

7680 rows × 3 columns

In [5]:
```
# Preprocessing functions
```

In [6]:
```
# Date De-Identification and URL Removal
def remove_mentions_and_url(text):
    mention_pattern = re.compile(r'@\w+')
    url_pattern = re.compile(r'https?://\S+|www\.\S+')

    # Use re.sub to remove mentions
    cleaned_text = mention_pattern.sub('', text)

    # Use re.sub to replace URLs with an empty string
    cleaned_text = url_pattern.sub('', cleaned_text)

    # Remove extra spaces and strip leading/trailing spaces
    cleaned_text = ' '.join(cleaned_text.split())

    return cleaned_text
```

```
# Special Character Removal
def remove_special_characters(text):
    text = emoji.replace_emoji(text, replace="[emoji]")

    # Split the text into words
    words = text.split(" ")

    # Initialize an empty string to store the cleaned text
    cleaned_text = ""

    # Iterate through each word
    for word in words:
        # Check if the word contains only special characters or "[emoji]"
        if not (re.match(r"^[_\W]+$", word) or "[emoji]" in word):
            if len(cleaned_text) == 0:
                cleaned_text = f"{word}"
            else:
                cleaned_text = f"{cleaned_text} {word}"

    # Remove diacritics
    text_no_diacritics = unidecode.unidecode(cleaned_text)

    # Split the text into words
    sentence = text_no_diacritics.split(" ")
    output = ""

    # Remove special characters and numerics
    for part in sentence:
        part = re.sub("[^A-Za-z ]+$", "", part)
        part = re.sub("^[^A-Za-z #]+", "", part)
        if not (len(part) <= 1 or re.match(r"[^a-zA-Z#]", part)):
            if len(output) == 0:
                output = f"{part}"
            else:
                output = f"{output} {part}"

    # Remove extra spaces and strip leading/trailing spaces
    cleaned_text = ' '.join(output.split())

    return cleaned_text
```

```python
# Hashtag Removal
def remove_hashtags(text):
    # Split the text into words
    words = text.split()

    # Initialize an empty list to store cleaned words
    cleaned_words = []

    for word in words:
        # Check if the word is a hashtag (starts with #)
        if not word.startswith('#'):
            cleaned_words.append(word)

    # Join the cleaned words into a single string
    cleaned_text = ' '.join(cleaned_words)

    return cleaned_text

# Lowercase except ALL-CAPS
def lowercase_except_all_caps(text):
    cleaned_text = remove_special_characters(text)

    words = cleaned_text.split()
    filtered_words = []

    for word in words:
        if word.isupper() and not word.istitle():
            filtered_words.append(word)
        else:
            filtered_words.append(re.sub(r'([A-Z][a-z]+)', lambda x: x.group(1).lower(), word))

    return ' '.join(filtered_words)

# Tokenization
def preprocess_text(text):
    tokens = tokenizer.tokenize(text)
    tokens = tokens[:tokenizer.model_max_length - 2]  # Account for [CLS] and [SEP] tokens
    indexed_tokens = tokenizer.convert_tokens_to_ids(tokens)
    return indexed_tokens
```

In [7]:
```python
for i in range(10):
    text = data_df["text"][i]
    label = data_df["label"][i]

    print('Text: ', text, "\n\nLabel: ", label, "\n\n----------------------------------------------------\n")
```

In [8]:
```python
# Preprocessing
```

In [9]:
```python
# Data De-Identification and URL Removal
data_df['text'] = data_df['text'].apply(remove_mentions_and_url)

for i in range(10):
    text = data_df["text"][i]
    label = data_df["label"][i]

    print('Text: ', text, "\n\nLabel: ", label, "\n\n----------------------------------------------------\n")
```

In [11]:
```python
# Special Character Removal
data_df['text'] = data_df['text'].apply(remove_special_characters)

for i in range(10):
    text = data_df["text"][i]
    label = data_df["label"][i]

    print('Text: ', text, "\n\nLabel: ", label, "\n\n----------------------------------------------------\n")
```

Hashtags Removed:

In [13]:
```python
# Hashtag Removal
data_df['text'] = data_df['text'].apply(remove_hashtags)

for i in range(10):
    text = data_df["text"][i]
    label = data_df["label"][i]

    print('Text: ', text, "\n\nLabel: ", label, "\n\n----------------------------------------------------\n")
```

Hashtags Retained:

In [13]:
```
# Hashtag Removal
# data_df['text'] = data_df['text'].apply(remove_hashtags)

# for i in range(10):
#     text = data_df["text"][i]
#     label = data_df["label"][i]

#     print('Text: ', text, "\n\nLabel: ", label, "\n\n----------------------------------------------------\n")
```

Normalization (Lowercase):

In [14]:
```
# Normalization - Lowercase
data_df['text'] = data_df['text'].str.lower()

# Lowercase Except ALL-CAPS
# data_df['text'] = data_df['text'].apply(lowercase_except_all_caps)

for i in range(10):
    text = data_df["text"][i]
    label = data_df["label"][i]

    print('Text: ', text, "\n\nLabel: ", label, "\n\n----------------------------------------------------\n")
```

Normalization (Lowercase except ALL-CAPS):

In [14]:
```
# Normalization - Lowercase
data_df['text'] = data_df['text'].str.lower()

# Lowercase Except ALL-CAPS
# data_df['text'] = data_df['text'].apply(lowercase_except_all_caps)

for i in range(10):
    text = data_df["text"][i]
    label = data_df["label"][i]

    print('Text: ', text, "\n\nLabel: ", label, "\n\n----------------------------------------------------\n")
```

In [15]:
```
data_df.to_csv('fbinary-7030.csv', index=False)
```

In [16]:
```
data_df
```

Out[16]:

	text	sentiment	label
0	absent pa rin sa comelec-sponsored presidentia...	Negative	Hate
1	in the philippines presidential frontrunner is...	Negative	Hate
2	the deluded one filipino politician who claims...	Negative	Hate
3	walang kaaway at hindi nang-aaway si matino at...	Negative	Hate
4	the young ones they don't know how dictator's ...	Negative	Hate
...	...	...	...
5115	look dating sen nakikipagpulong sa kaniyang te...	Neutral	Non-hate
5116	gagi nahihiwagaan talaga ako sa mga atabs na l...	Neutral	Non-hate
5117	manifesting circle as president of the philipp...	Neutral	Non-hate
5118	ang tatanglaw sa ating bayan ay isang ilaw ng ...	Neutral	Non-hate
5119	on undocumented ofws president has these to sa...	Neutral	Non-hate

5120 rows × 3 columns

In [17]:
```
data_df['text'] = data_df['text'].apply(preprocess_text)
```

In [18]:
```
data_df.to_csv('fbinary-tokenized-7030.csv', index=False)
```

In [19]:
```
data_df
```

Out[19]:

	text	sentiment	label
0	[9962, 6643, 15544, 2078, 7842, 2272, 2571, 22...	Negative	Hate
1	[1999, 1996, 5137, 4883, 2392, 23195, 2003, 24...	Negative	Hate
2	[1996, 3972, 13936, 2028, 10275, 3761, 2040, 4...	Negative	Hate
3	[24547, 5654, 10556, 9497, 2012, 9269, 16660, ...	Negative	Hate
4	[1996, 2402, 3924, 2027, 2123, 1005, 1056, 211...	Negative	Hate
...	...	...	...
5115	[2298, 5306, 12411, 17823, 17471, 4502, 21600,...	Neutral	Non-hate
5116	[18201, 2072, 20976, 19190, 2072, 4213, 3654, ...	Neutral	Non-hate
5117	[19676, 2075, 4418, 2004, 2343, 1997, 1996, 5137]	Neutral	Non-hate
5118	[17076, 23236, 3070, 14919, 7842, 2012, 2075, ...	Neutral	Non-hate
5119	[2006, 25672, 24894, 14088, 1997, 9333, 2343, ...	Neutral	Non-hate

5120 rows × 3 columns

70:30

In [20]:
```
train_df, test_df = train_test_split(data_df, test_size=0.3, random_state=SEED)
```

80:20

In [20]:
```
train_df, test_df = train_test_split(data_df, test_size=0.2, random_state=SEED)
```

90:10

In [20]:
```
train_df, test_df = train_test_split(data_df, test_size=0.1, random_state=SEED)
```

Binary:

In [21]:
```python
import torch.nn as nn
import torch.nn.functional as F

class CNN(nn.Module):
    def __init__(self, vocab_size, embedding_dim, n_filters, filter_sizes, output_dim, dropout):
        super().__init__()

        self.embedding = nn.Embedding(vocab_size, embedding_dim)
        self.embedding_dropout = nn.Dropout(dropout)

        # Convolutional layers with varying kernel sizes
        self.convs = nn.ModuleList([
            nn.Conv1d(in_channels=embedding_dim, out_channels=n_filters, kernel_size=fs, padding='same')
            for fs in filter_sizes
        ])

        # Batch normalization for each convolutional layer
        self.bns = nn.ModuleList([nn.BatchNorm1d(n_filters) for _ in filter_sizes])

        # Global Max Pooling
        self.global_pooling = nn.AdaptiveMaxPool1d(1)

        # Dropout
        self.dropout = nn.Dropout(dropout)

        # Fully connected layer
        self.fc = nn.Linear(len(filter_sizes) * n_filters, output_dim)

    def forward(self, x):
        embedded = self.embedding(x)
        embedded = self.embedding_dropout(embedded)
        x = embedded.permute(0, 2, 1)

        # Apply convolutional and batch normalization layers
        x = [F.relu(bn(conv(x))) for conv, bn in zip(self.convs, self.bns)]
        x = [self.global_pooling(conv).squeeze(2) for conv in x]

        # Concatenate the feature maps
        x = torch.cat(x, 1)

        x = self.dropout(x)
        x = self.fc(x)

        return x
```

Multi Label:

In [21]:
```python
import torch.nn as nn
import torch.nn.functional as F

class CNN(nn.Module):
    def __init__(self, vocab_size, embedding_dim, n_filters, filter_sizes, num_classes, dropout):
        super().__init__()

        self.embedding = nn.Embedding(vocab_size, embedding_dim)
        self.embedding_dropout = nn.Dropout(dropout)

        # Convolutional layers with varying kernel sizes
        self.convs = nn.ModuleList([
            nn.Conv1d(in_channels=embedding_dim, out_channels=n_filters, kernel_size=fs, padding='same')
            for fs in filter_sizes
        ])

        # Batch normalization for each convolutional layer
        self.bns = nn.ModuleList([nn.BatchNorm1d(n_filters) for _ in filter_sizes])

        # Global Max Pooling
        self.global_pooling = nn.AdaptiveMaxPool1d(1)

        # Dropout for regularization
        self.dropout = nn.Dropout(dropout)

        # Fully connected layer for classification
        self.fc = nn.Linear(len(filter_sizes) * n_filters, num_classes)

    def forward(self, x):
        embedded = self.embedding(x)
        embedded = self.embedding_dropout(embedded)
        x = embedded.permute(0, 2, 1)

        # Apply each convolutional and batch normalization layer
        x = [F.relu(bn(conv(x))) for conv, bn in zip(self.convs, self.bns)]
        x = [self.global_pooling(conv).squeeze(2) for conv in x]

        # Concatenate the feature maps
        x = torch.cat(x, 1)

        x = self.dropout(x)
        x = self.fc(x)

        return x
```

In [22]:
```python
# Set up iterators
BATCH_SIZE = 64
```

In [23]:
```python
class TextDataset(torch.utils.data.Dataset):
    def __init__(self, dataframe, max_seq_length):
        self.data = dataframe
        self.max_seq_length = max_seq_length

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        text = self.data.iloc[idx]['text']
        label = self.data.iloc[idx]['label']

        # Padding and conversion to tensor
        padded_text = torch.tensor(text[:self.max_seq_length] + [0] * (self.max_seq_length - len(text)))
        return padded_text, label
```

In [24]:
```python
train_dataset = TextDataset(train_df, 1000)
test_dataset = TextDataset(test_df, 1000)
```

In [25]:
```python
train_iterator = torch.utils.data.DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle=True)
test_iterator = torch.utils.data.DataLoader(test_dataset, batch_size=BATCH_SIZE, shuffle=False)
```

In [26]:
```python
# Define model hyperparameters
VOCAB_SIZE = bert_model.config.vocab_size
EMBEDDING_DIM = bert_model.config.hidden_size
N_FILTERS = 100
FILTER_SIZES = [3, 4, 5]
OUTPUT_DIM = 1
DROPOUT = 0.5
```

Binary:

In [27]:
```
# Initialize CNN model
model = CNN(VOCAB_SIZE, EMBEDDING_DIM, N_FILTERS, FILTER_SIZES, OUTPUT_DIM, DROPOUT)
model.embedding.weight.data.copy_(bert_model.embeddings.word_embeddings.weight.data)
```

Out[27]:
```
tensor([[ 0.0000e+00,  0.0000e+00,  0.0000e+00,  ...,  0.0000e+00,
          0.0000e+00,  0.0000e+00],
        [ 1.3477e-02,  4.2843e-02, -3.2582e-03,  ...,  4.9621e-03,
          9.0322e-03,  5.5630e-05],
        [-2.3741e-02, -2.9270e-02, -1.0655e-02,  ..., -1.0585e-02,
         -7.0327e-03,  1.2088e-02],
        ...,
        [ 1.8732e-02, -1.8530e-02,  2.3105e-02,  ..., -6.3155e-03,
          2.7844e-02,  2.3055e-02],
        [-5.7605e-03, -4.5277e-03,  1.1638e-02,  ...,  7.5517e-03,
          5.6461e-03,  7.8079e-03],
        [ 2.9906e-02,  6.0971e-03,  5.6574e-03,  ..., -4.1868e-02,
         -1.1391e-02, -3.8559e-02]])
```

In [28]:
```
# Initialize BERT model (for embedding extraction)
bert_model.eval()  # Set to evaluation mode
```

Out[28]:
```
BertModel(
  (embeddings): BertEmbeddings(
    (word_embeddings): Embedding(30522, 1024, padding_idx=0)
    (position_embeddings): Embedding(512, 1024)
    (token_type_embeddings): Embedding(2, 1024)
    (LayerNorm): LayerNorm((1024,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.9, inplace=False)
  )
  (encoder): BertEncoder(
    (layer): ModuleList(
      (0-15): 16 x BertLayer(
        (attention): BertAttention(
          (self): BertSelfAttention(
            (query): Linear(in_features=1024, out_features=1024, bias=True)
            (key): Linear(in_features=1024, out_features=1024, bias=True)
            (value): Linear(in_features=1024, out_features=1024, bias=True)
            (dropout): Dropout(p=0.9, inplace=False)
          )
          (output): BertSelfOutput(
            (dense): Linear(in_features=1024, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.9, inplace=False)
          )
        )
        (intermediate): BertIntermediate(
          (dense): Linear(in_features=1024, out_features=4096, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): BertOutput(
          (dense): Linear(in_features=4096, out_features=1024, bias=True)
          (LayerNorm): LayerNorm((1024,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.9, inplace=False)
        )
      )
    )
  )
  (pooler): BertPooler(
    (dense): Linear(in_features=1024, out_features=1024, bias=True)
    (activation): Tanh()
  )
)
```

Multi Label:

In [27]:
```
# Initialize CNN model
model = CNN(VOCAB_SIZE, EMBEDDING_DIM, N_FILTERS, FILTER_SIZES, 3, DROPOUT)
model.embedding.weight.data.copy_(bert_model.embeddings.word_embeddings.weight.data)
```

Out[27]:
```
tensor([[ 0.0000e+00,  0.0000e+00,  0.0000e+00,  ...,  0.0000e+00,
          0.0000e+00,  0.0000e+00],
        [-4.0194e-02, -2.5010e-03,  1.4670e-02,  ...,  1.4063e-02,
          9.1368e-03, -2.1431e-02],
        [-1.7789e-02, -1.2694e-02,  3.4012e-02,  ...,  1.9803e-02,
          1.7356e-02, -4.6034e-02],
        ...,
        [ 5.8626e-03, -1.0221e-02,  2.4332e-02,  ...,  4.3287e-02,
          2.8594e-02, -2.1989e-04],
        [ 1.8230e-02, -2.2223e-02, -7.4849e-03,  ..., -8.8552e-05,
          8.9816e-04,  6.9264e-03],
        [ 5.8198e-02, -5.4685e-03,  2.4872e-02,  ..., -3.3694e-02,
         -9.1587e-03,  1.3332e-02]])
```

In [28]:
```
# Initialize BERT model (for embedding extraction)
bert_model.eval()  # Set to evaluation mode
```

Out[28]:
```
BertModel(
  (embeddings): BertEmbeddings(
    (word_embeddings): Embedding(30522, 1024, padding_idx=0)
    (position_embeddings): Embedding(512, 1024)
    (token_type_embeddings): Embedding(2, 1024)
    (LayerNorm): LayerNorm((1024,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.9, inplace=False)
  )
  (encoder): BertEncoder(
    (layer): ModuleList(
      (0-15): 16 x BertLayer(
        (attention): BertAttention(
          (self): BertSelfAttention(
            (query): Linear(in_features=1024, out_features=1024, bias=True)
            (key): Linear(in_features=1024, out_features=1024, bias=True)
            (value): Linear(in_features=1024, out_features=1024, bias=True)
            (dropout): Dropout(p=0.9, inplace=False)
          )
          (output): BertSelfOutput(
            (dense): Linear(in_features=1024, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.9, inplace=False)
          )
        )
        (intermediate): BertIntermediate(
          (dense): Linear(in_features=1024, out_features=4096, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): BertOutput(
          (dense): Linear(in_features=4096, out_features=1024, bias=True)
          (LayerNorm): LayerNorm((1024,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.9, inplace=False)
        )
      )
    )
  )
  (pooler): BertPooler(
    (dense): Linear(in_features=1024, out_features=1024, bias=True)
    (activation): Tanh()
  )
)
```

In [29]:
```
with torch.no_grad():
    for i, token in enumerate(tokenizer.get_vocab()):
        token_id = tokenizer.convert_tokens_to_ids(token)
        token_embedding = bert_model.embeddings.word_embeddings.weight[token_id]
        model.embedding.weight[i].data.copy_(token_embedding)

bert_parameters = []
for layer in bert_model.encoder.layer:
    bert_parameters.extend(layer.parameters())

# Create AdamW optimizer with custom hyperparameters for BERT embeddings
bert_learning_rate = 2e-5  # Adjust as needed
bert_optimizer = optim.AdamW(bert_parameters, lr=bert_learning_rate)
```

In [30]:
```
#optimizer = optim.Adam(model.parameters())
# Your custom hyperparameters
learning_rate = 0.001
beta_1 = 0.9
beta_2 = 0.999
epsilon = 1e-08
weight_decay = 0.0

# Create Adam optimizer with custom hyperparameters
optimizer = optim.Adam(model.parameters(), lr=learning_rate, betas=(beta_1, beta_2), eps=epsilon, weight_decay=weight_decay)
criterion = nn.BCEWithLogitsLoss()
```

Binary:

In [31]:
```python
# Train function
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

def train(model, iterator):
    model.train()
    epoch_loss = 0
    for text_batch, label_batch in iterator:
        # Extract text sequences from the text_batch tensor
        texts = text_batch

        # Extract and process labels
        labels = [1 if label == 'Hate' else 0 for label in label_batch]  # Example conversion

        texts = texts.to(device)  # Move to device if needed
        labels = torch.tensor(labels, dtype=torch.float32).to(device)  # Convert to tensor

        optimizer.zero_grad()
        predictions = model(texts).squeeze(1)

        loss = criterion(predictions, labels)
        loss.backward()
        optimizer.step()
        epoch_loss += loss.item()
    return epoch_loss / len(iterator)
```

```python
# Evaluate function
def evaluate(model, iterator):
    model.eval()
    epoch_loss = 0
    predicted_labels = []
    true_labels = []  # Declare the true_labels list

    with torch.no_grad():
        for text_batch, label_batch in iterator:
            texts = text_batch  # Extract text sequences
            labels = [1 if label == 'Hate' else 0 for label in label_batch]  # Example conversion

            texts = texts.to(device)  # Move to device
            labels = torch.tensor(labels, dtype=torch.float32).to(device)  # Convert to tensor

            predictions = model(texts).squeeze(1)
            loss = criterion(predictions, labels)

            epoch_loss += loss.item()
            predicted_labels.extend(torch.round(torch.sigmoid(predictions)).cpu().numpy())
            true_labels.extend(labels.cpu().numpy())

    # Calculate accuracy, f1, precision, recall
    accuracy = accuracy_score(true_labels, predicted_labels)
    f1 = f1_score(true_labels, predicted_labels)
    precision = precision_score(true_labels, predicted_labels)
    recall = recall_score(true_labels, predicted_labels)

    return epoch_loss / len(iterator), accuracy, f1, precision, recall, predicted_labels, true_labels
```

Multi Label:

In [31]:
```python
# Create a dictionary to map class labels to numerical indices
label_to_index = {
    "Positive": 0,
    "Negative": 1,
    "Neutral": 2
}

# Train function
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

def train(model, iterator, criterion, optimizer, device):
    model.train()
    epoch_loss = 0
    for text_batch, sentiment_batch in iterator:
        texts = text_batch.to(device)
        labels = torch.tensor([label_to_index[sentiment] for sentiment in sentiment_batch], dtype=torch.long).to(device)

        optimizer.zero_grad()
        predictions = model(texts)  # Removed squeeze(1)
        loss = criterion(predictions, labels)
        loss.backward()
        optimizer.step()

        epoch_loss += loss.item()

    return epoch_loss / len(iterator)
```

```python
# Evaluate function
def evaluate(model, iterator):
    model.eval()
    epoch_loss = 0
    predicted_sentiments = []
    true_sentiments = []  # Declare the true_sentiments list

    with torch.no_grad():
        for text_batch, sentiment_batch in iterator:
            texts = text_batch  # Extract text sequences

            # Use numerical labels
            labels = torch.tensor([label_to_index[sentiment] for sentiment in sentiment_batch]).to(device)

            texts = texts.to(device)  # Move to device

            predictions = model(texts).squeeze(1)
            loss = criterion(predictions, labels)

            epoch_loss += loss.item()

            # Convert predictions to class labels
            predicted_sentiments.extend([list(label_to_index.keys())[list(label_to_index.values()).index(idx)] for idx in torch.a
            true_sentiments.extend([list(label_to_index.keys())[list(label_to_index.values()).index(idx)] for idx in labels.cpu()

    # Calculate accuracy, f1, precision, recall
    accuracy = accuracy_score(true_sentiments, predicted_sentiments)
    f1 = f1_score(true_sentiments, predicted_sentiments, average='macro')
    precision = precision_score(true_sentiments, predicted_sentiments, average='macro')
    recall = recall_score(true_sentiments, predicted_sentiments, average='macro')

    return epoch_loss / len(iterator), accuracy, f1, precision, recall, predicted_sentiments, true_sentiments
```

Binary:

In [32]:
```
from sklearn.metrics import classification_report
import os

best_test_loss = float('inf')
best_epoch = 0

N_EPOCHS = 10
predicted_labels_per_epoch = []
true_labels_per_epoch = []

for epoch in range(N_EPOCHS):
    train_loss = train(model, train_iterator)
    test_loss, accuracy, f1, precision, recall, predicted_labels, true_labels = evaluate(model, test_iterator)

    # Save the predicted_labels and true_labels for the current epoch
    predicted_labels_per_epoch.append(predicted_labels)
    true_labels_per_epoch.append(true_labels)

    # Print other metrics (e.g., accuracy, loss) for the current epoch
    print(f'Epoch: {epoch + 1:02}\n')
    print(f'\tTrain Loss: {train_loss:.3f}')
    print(f'\tTest Loss: {test_loss:.3f}\n')
    print(f'\tAccuracy: {accuracy:.4f} | F1-Score: {f1:.4f}')
    print(f'\tPrecision: {precision:.4f} | Recall: {recall:.4f}\n')
    print(f'Classification Report:\n')
    report = classification_report(true_labels, predicted_labels, target_names=["0", "1"], digits=4)
    print(f'{report}\n')
```

```
    from sklearn.metrics import confusion_matrix
    import seaborn as sns
    import matplotlib.pyplot as plt
    import numpy as np  # Import numpy for matrix normalization

    # Calculate the confusion matrix
    cf_matrix = confusion_matrix(true_labels, predicted_labels)
    print(cf_matrix)

    # Plot the confusion matrix using seaborn
    sns.set(rc={"figure.figsize": (8, 6)})
    ax = sns.heatmap(cf_matrix / np.sum(cf_matrix), annot=True, xticklabels=['Non-Hate', 'Hate'], yticklabels=['Non-Hate', 'Hate'

    plt.title('Confusion Matrix\n(Binary BERT-CNN 70:30)\n', fontsize=13)
    plt.show()

    if test_loss <= best_test_loss:
        best_test_loss = test_loss
        best_epoch = epoch

        # Save BERT model
        torch.save(bert_model.state_dict(), 'THESIS-BINARY-7030-BERT.pt')

        # Save CNN model
        torch.save(model.state_dict(), 'THESIS-BINARY-7030-CNN.pt')

# Print information about the best epoch
print(f"Best Model saved at epoch {best_epoch + 1} with Test Loss: {best_test_loss:.3f}")
```

Out[33]:

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
CNN                                      [64, 1]                   --
├─Embedding: 1-1                         [64, 128, 1024]           31,254,528
├─Dropout: 1-2                           [64, 128, 1024]           --
├─ModuleList: 1-7                        --                        (recursive)
│    └─Conv1d: 2-1                       [64, 100, 128]            307,300
├─ModuleList: 1-8                        --                        (recursive)
│    └─BatchNorm1d: 2-2                  [64, 100, 128]            200
├─ModuleList: 1-7                        --                        (recursive)
│    └─Conv1d: 2-3                       [64, 100, 128]            409,700
├─ModuleList: 1-8                        --                        (recursive)
│    └─BatchNorm1d: 2-4                  [64, 100, 128]            200
├─ModuleList: 1-7                        --                        (recursive)
│    └─Conv1d: 2-5                       [64, 100, 128]            512,100
├─ModuleList: 1-8                        --                        (recursive)
│    └─BatchNorm1d: 2-6                  [64, 100, 128]            200
├─AdaptiveMaxPool1d: 1-9                 [64, 100, 1]              --
├─AdaptiveMaxPool1d: 1-10                [64, 100, 1]              --
├─AdaptiveMaxPool1d: 1-11                [64, 100, 1]              --
├─Dropout: 1-12                          [64, 300]                 --
├─Linear: 1-13                           [64, 1]                   301
==========================================================================================
Total params: 32,484,529
Trainable params: 32,484,529
Non-trainable params: 0
Total mult-adds (G): 12.07
==========================================================================================
Input size (MB): 0.07
Forward/backward pass size (MB): 106.43
Params size (MB): 129.94
Estimated Total Size (MB): 236.43
==========================================================================================
```

Multi Label:

In [32]:
```
from sklearn.metrics import classification_report
import os

best_test_loss = float('inf')
best_epoch = 0

N_EPOCHS = 10
predicted_sentiments_per_epoch = []
true_sentiments_per_epoch = []

for epoch in range(N_EPOCHS):
    train_loss = train(model, train_iterator, criterion, optimizer, device)
    test_loss, accuracy, f1, precision, recall, predicted_sentiments, true_sentiments = evaluate(model, test_iterator)

    # Save the predicted_sentiments and true_sentiments for the current epoch
    predicted_sentiments_per_epoch.append(predicted_sentiments)
    true_sentiments_per_epoch.append(true_sentiments)

    # Print other metrics (e.g., accuracy, loss) for the current epoch
    print(f'Epoch: {epoch + 1:02}\n')
    print(f'\tTrain Loss: {train_loss:.3f}')
    print(f'\tTest Loss: {test_loss:.3f}\n')
    print(f'\tAccuracy: {accuracy:.4f} | F1-Score: {f1:.4f}')
    print(f'\tPrecision: {precision:.4f} | Recall: {recall:.4f}\n')
    print(f'Classification Report:\n')
    report = classification_report(true_sentiments, predicted_sentiments, target_names=['Positive', 'Negative', 'Neutral'], digit
    print(f'{report}\n')
```

```
    from sklearn.metrics import confusion_matrix
    import seaborn as sns
    import matplotlib.pyplot as plt
    import numpy as np  # Import numpy for matrix normalization

    # Calculate the confusion matrix
    cf_matrix = confusion_matrix(true_sentiments, predicted_sentiments)
    print(cf_matrix)

    # Plot the confusion matrix using seaborn
    sns.set(rc={"figure.figsize": (8, 6)})  # Add a closing parenthesis to set
    ax = sns.heatmap(cf_matrix / np.sum(cf_matrix), annot=True, xticklabels=['Positive', 'Negative', 'Neutral'], yticklabels=['Po

    plt.title('Confusion Matrix\n(Multilabel BERT-CNN 70:30)\n', fontsize=13)
    plt.show()

    if test_loss <= best_test_loss:
        best_test_loss = test_loss
        best_epoch = epoch

        # Save BERT model
        torch.save(bert_model.state_dict(), 'THESIS-MULTI-7030-BERT.pt')

        # Save CNN model
        torch.save(model.state_dict(), 'THESIS-MULTI-7030-CNN.pt')

# Print information about the best epoch
print(f"Best Model saved at epoch {best_epoch + 1} with Test Loss: {best_test_loss:.3f}")
```

In [37]:
```
from torchinfo import summary

sample_input = torch.zeros((BATCH_SIZE, max_seq_length), dtype=torch.long)
summary(model, input_data=sample_input)
```

Out[37]:
```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
CNN                                      [64, 3]                   --
├─Embedding: 1-1                         [64, 128, 1024]           31,254,528
├─Dropout: 1-2                           [64, 128, 1024]           --
├─ModuleList: 1-7                        --                        (recursive)
│    └─Conv1d: 2-1                       [64, 100, 128]            307,300
├─ModuleList: 1-8                        --                        (recursive)
│    └─BatchNorm1d: 2-2                  [64, 100, 128]            200
├─ModuleList: 1-7                        --                        (recursive)
│    └─Conv1d: 2-3                       [64, 100, 128]            409,700
├─ModuleList: 1-8                        --                        (recursive)
│    └─BatchNorm1d: 2-4                  [64, 100, 128]            200
├─ModuleList: 1-7                        --                        (recursive)
│    └─Conv1d: 2-5                       [64, 100, 128]            512,100
├─ModuleList: 1-8                        --                        (recursive)
│    └─BatchNorm1d: 2-6                  [64, 100, 128]            200
├─AdaptiveMaxPool1d: 1-9                 [64, 100, 1]              --
├─AdaptiveMaxPool1d: 1-10                [64, 100, 1]              --
├─AdaptiveMaxPool1d: 1-11                [64, 100, 1]              --
├─Dropout: 1-12                          [64, 300]                 --
├─Linear: 1-13                           [64, 3]                   903
==========================================================================================
Total params: 32,485,131
Trainable params: 32,485,131
Non-trainable params: 0
Total mult-adds (Units.GIGABYTES): 12.07
==========================================================================================
Input size (MB): 0.07
Forward/backward pass size (MB): 106.43
Params size (MB): 129.94
Estimated Total Size (MB): 236.44
==========================================================================================
```

Appendix M

Researcher's Curriculum Vitae

Micah Collette O. Mendoza

Block 116 Lot 33, Angelica Drive, Robinsons Homes East,
Brgy. San Jose, Antipolo City, Rizal
(+63) 9281794483
micahcollette.mendoza.cics@ust.edu.ph

Education

UNIVERSITY OF SANTO TOMAS
Bachelor of Computer Science with Specialization in Data Science
2020 – Present

UNIVERSITY OF THE PHILIPPINES BAGUIO
Bachelor of Arts in Social Sciences, Major in History, Minor in Philosophy
2019 – 2020

LA SALLE COLLEGE ANTIPOLO
Senior High School
2017-2019

DURAT AL-SHARQ INTERNATIONAL SCHOOL
Junior High School
2013-2017

Programming Experience

NOYPEE: Local Indulgence at your Doorstep!

- Developed and maintained the back-end infrastructure for NOYPEE, a local e-commerce platform showcasing Philippine products.
- Implemented critical features such as user authentication, password modification, and report generation, enhancing overall application functionality.
- Collaborated with front-end developers to integrate interfaces, ensuring a seamless and user-friendly e-commerce experience.
- Utilized a local database for efficient data management, including product information, orders, and user data.

Seminars

- **DLSU Technology Summit 2020 "Mapping Future Possibilities"** (2020)
- **Vuja De Tech Talks** (2021)
- **AceIT: UI/UX Strategies Webinar** (2021)
- **JavaScript Workshop** (2021)
- **ARISE: Community Development Webinar** (2021)
- **Oracle Netsuite Tech Talks** (2021)
- **Vlogger's Webinar: Social Media Etiquette and Video Blogging** (2021)
- **Hello World: Meet ML and VR** (2021)
- **Designing the Future: Navigating through the World of UX and Tech** (2021)
- **Increased Productivity and Boost Security Microsoft Office 365** (2021)
- **CrypTalk: Currency and NFTs** (2022)
- **Salig: How to Keep an Active Faith** (2022)
- **Python Workshop (2022)**
- **Vuja De Tech Talks** (2023)
- **CS Research Colloquium** (2023)

Technical Skills

- **Languages:** Python, Java, SQL, HTML5, CSS, Dart
- **Frameworks:** Flutter
- **Developer Tools:** Visual Studio Code, Jupyter Notebook, Android Studio, Apache NetBeans, GitHub
- **Database:** MySQL, PostgreSQL
- **Operating System:** Windows

Wayne Gabriel S. Nadurata

Unit No. 408, Building 224 Vizione Residences, 2nd St., Barangay 11
1403, Caloocan City, Philippines
(+63) 09326459064
waynegabriel.nadurata.cics@ust.edu.ph

Education

UNIVERSITY OF SANTO TOMAS, Manila
Bachelor of Computer Science with Specialization in Data Science
2020 - Present

UNIVERSITY OF SANTO TOMAS, Manila
Senior High School
2018-2020

LA CONSOLACION COLLEGE - CALOOCAN, Manila
Junior High School
2014-2018

Programming Experience

COMPUTER SCIENCE SOCIETY, University of Santo Tomas (2023 – Present)

- Programmed and implemented the front end and back end design for the Computer Science Society website, particularly with the news and events page.
- Acquire the skill to code using React JS, SCSS and Next JS.
- Learn the skill to update datasets and maintain websites.
- Learn the skill to link social media posts to websites using JSON Files.

CADA KAPE, Taft, Metro Manila (2023 - Present)

- Programmed an employee attendance sheet with encoded Late/Overtime/Undertime status per clock in and clock out.
- Improved the skill to code, compose and maintain spreadsheets of data using Microsoft Excel.

Leadership Experience

COMPUTER SCIENCE SOCIETY, University of Santo Tomas (2023 – Present)

- Deliberate new Computer Science Society web developer members.
- Maintain and update the website consistently with the latest news and events organized and produced by the Computer Science Society organization.
- Secure the webpage for bugs and technical problems.

Technical Skills

- **Programming:** Java, Python, HTML5, ReactJS, NextJS, Cascading Style Sheets (CSS), Sassy Cascading Style Sheets (SCSS) MySQL, R, Flutter/Dart.
- **Software:** Apache Netbeans, Jupyter Notebook, R Studio, Android Studio, Visual Studio Code, GitHub and Microsoft Excel.
- **Operating System:** Windows and MAC OS.
- **Web Programming:** Co-Developed the website www.ust-css.com

Graphics Skills

- **Software:** Wacom, Da Vinci Resolve Studios, Canva, FireAlpaca, Adobe Photoshop, ibisPaintX
- **Operating System:** Windows.
- **Video Editing:** Practice editing on my Youtube channel www.youtube.com/@bainiku

Mark Gabriel E. Ortiz

08 – 033 Bagong Lipunan Condominium, Western Bicutan,
Taguig City, Philippines
(+63) 09490406756
waynegabriel.nadurata.cics@ust.edu.ph

Education

UNIVERSITY OF SANTO TOMAS, Manila
Bachelor of Computer Science with Specialization in Data Science
2020 - Present

DON BOSCO TECHNICAL INSITUTE, Makati
Senior High School
2018-2020

DON BOSCO TECHNICAL INSTITUTE, Makati
Junior High School
2014-2018

Programming Experience

TAGBAKIN HARDWARE INVENTORY SYSTEM (2022 - 2023)

- Programmed and implemented the back-end functionality of the Tagbakin Hardware Inventory System which is coded using Flutter as a framework and NoSql through Google Firestore

UST EVOSYS (2022-present)

- Currently one of the programmers of the UST EvoSys which is used during local, and university-wide elections.

Leadership Experience

COMPUTER SCIENCE SOCIETY, University of Santo Tomas (2020 – Present)

- Assistant Secretary and one of the committee heads of the Logistics and Technology Development Committee
- Co-project head of the CSS General Assembly 2023
- Staff of CodeSprint 2023, CS Research Colloquium 2023, Meet the Parents 2023, Vuja De 2023, CS Fundamentals 2023
- Former Executive Associate to the Secretary (2022 – 2023)
- Former Technology Development Committee Staff (2021 – 2022)
- Former Academics Committee Staff (2020-2021)

Technical Skills

- **Programming:** Java, Python, C#, ASP.net, MySQL, R, Flutter/Dart, NoSQL, Postgre.
- **Software:** Apache Netbeans, Jupyter Notebook, R Studio, Android Studio, Visual Studio Code, Visual Studio, GitHub
- **Operating System:** Windows

Joshua Mari L. Padlan

688 Malinis Street Riverside Subdivision Barangay Santo Domingo Cainta Rizal
(+63) 09082944238
Joshuamari.padlan.cics@ust.edu.ph

Education

UNIVERSITY OF SANTO TOMAS, Manila
Bachelor of Computer Science
2015 - Present

San Beda College, Rizal
High School
2011-2015

Sienna College, Taytay
Elementary School
2005-2011

Notable Achievements

- **IT Certification Exam Passer** - 2021 October 23

Technical Skills

- **Programming:** C++, Java, Python, HTML.
- **Software:** MySQL, Apache Netbeans, Android Studio, Visual Studio Code, GitHub and Microsoft Excel.
- **Operating System:** Windows.