Doctor of Philosophy in Computer Science and Engineering

Text Classification in a Resource-constrained Language using Deep Learning Techniques

by

Md. Rajib Hossain

ID: 18PCSE005P

Department of Computer Science and Engineering
CHITTAGONG UNIVERSITY OF ENGINEERING & TECHNOLOGY (CUET)
Chattogram-4349, Bangladesh.
February, 2024

Text Classification in a Resource-constrained Language using Deep Learning Techniques

Submitted for partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computer Science and Engineering

by
Md. Rajib Hossain
ID: 18PCSE005P

Supervised by:

Professor Dr. Mohammed Moshiul Hoque
Department of Computer Science and Engineering

Department of Computer Science and Engineering
CHITTAGONG UNIVERSITY OF ENGINEERING & TECHNOLOGY (CUET)
Chattogram-4349, Bangladesh.

Doctoral Committee

Dr. Mohammed Moshiul Hoque Professor, Dept. of CSE, CUET	Chairman (Supervisor)
Dr. Md. Iqbal Hasan Sarker Professor, Dept. of CSE, CUET	Co-Supervisor
Dr. Abu Hasnat Mohammad Ashfak Habib Professor, Dept. of CSE, CUET	Member (Ex-officio)
Dr. Muhammad Ibrahim Khan Professor, Dept. of CSE, CUET	Member (Internal)
Dr. Muhammad Ahsan Ullah Professor, Dept. of EEE, CUET	Member (Internal)
Dr. Asaduzzaman Professor, Dept. of CSE, CUET	Member (Internal)
Dr. Mohammad Shahidur Rahman Professor, Dept. of CSE, SUST	Member (External)

Declaration of Originality

I, Md. Rajib Hossain, solemnly affirm that this doctoral thesis, presented for the award of a PhD degree, is entirely my work. I further confirm that this thesis, in its entirety or part, has not been previously submitted to fulfill any degree at any other university or institution. I also declare that, to the best of my knowledge, this thesis does not infringe upon any copyright, intellectual property, or proprietary rights. Any ideas, theories, techniques, quotations, or other content from the works of others cited or referenced herein have been appropriately acknowledged in accordance with standard norms. Furthermore, I hereby transfer all rights in the copyright of this thesis to the Department of Computer Science and Engineering (CSE) at Chittagong University of Engineering and Technology (CUET). The department shall be the rightful owner of the copyright and any reproduction or utilization of this work in any form or by any means is strictly prohibited without the prior consent of the Department of CSE, CUET.

Signature of the Candidate

Md. Rajib Hossain
ID: 18PCSE005P
Session: 2018-2019
Department of Computer Science and Engineering
Chittagong University of Engineering & Technology (CUET)

Certification

The thesis titled **"Text Classification in a Resource-constrained Language using Deep Learning Techniques"** submitted by **Md. Rajib Hossain**, Roll No. **18PCSE005P**, Session **2018-2019** has been accepted as satisfactory in partial fulfillment of the requirement for the degree of **Doctor of Philosophy in Computer Science and Engineering** on 27 February 2024.

Board of Examiners

1. _____
Dr. Mohammed Moshiul Hoque — Chairman (Supervisor)
Professor, Dept. of CSE, CUET

2. _____
Dr. Md. Iqbal Hasan Sarker — Member (Co-supervisor)
Professor, Dept. of CSE, CUET

3. _____
Dr. Abu Hasnat Mohammad Ashfak Habib — Member (Ex-officio)
Professor, Dept. of CSE, CUET

4. _____
Dr. Muhammad Ibrahim Khan — Member (Internal)
Professor, Dept. of CSE, CUET

5. _____
Dr. Asaduzzaman — Member (Internal)
Professor, Dept. of CSE, CUET

6. _____
Dr. Md. Mokammel Haque — Member (Internal)
Professor, Dept. of CSE, CUET

7. _____
Dr. Mohammad Shahidur Rahman — Member (External)
Professor, Dept. of CSE, SUST

8. _____
Dr. Asif Ekbal — Member (External)
Associate Professor, Dept. of CSE, IIT Patna, India

Dedicated to,

my beloved parents,

....

my belobed wife, Sadia Afroze & daughter, Radia Hossain Rafida.

...

for their love, endless support, encouragement, and sacrifices.

Acknowledgment

I would like to express my sincere gratitude to my honorable supervisor, Professor Dr. Mohammed Moshiul Hoque, for his invaluable guidance, unwavering support, and scholarly insights throughout the doctoral study. His expertise, encouragement, and commitment to academic excellence have shaped this thesis. I am also profoundly thankful to the co-supervisor, Professor Dr. Iqbal H. Sarker, for his scholarly input, constructive feedback, and encouragement. His expertise in the field has significantly enriched the quality of this research.

We sincerely thank Dr. Nazmul Siddique, mentor and collaborator, for his valuable contributions, discussions, and collaborative efforts, particularly in [specific aspect of the research]. His insights and support have been instrumental in successfully completing this thesis. I thank the School of Computing, Engineering, and Intelligent Systems at Ulster University, Londonderry, UK, for providing a conducive research environment and necessary resources. Furthermore, we thank Dr. M Ali Akber Dewan from the School of Computing and Information Systems, Faculty of Science and Technology, Athabasca University, Alberta, Canada. His collaboration and contributions, particularly in [specific area of your research], have added depth and perspective to my research.

Finally, we wholeheartedly appreciate the Project 'Establishment of Sheikh Kamal IT Business Incubator at CUET,' BHTPA, and the ICT Division for their valuable support and cooperation. We also thank all the members of the CUET NLP Lab, faculty members, and staff of the department of CSE, CUET, for their dedicated support. My family deserves special thanks for their continuing love, support, encouragement, and sacrifices.

সংক্ষিপ্তসার

বিশ্বব্যাপী ওয়ার্ল্ড ওয়াইড ওয়েবে মানুষ এখন ব্যাপক ভাবে জড়িয়ে পড়ছে যেমন, সামাজিক যোগাযোগ মাধ্যম (ফেসবুক, ইউটিউব, এবং টুইটার), অনলাইন ভিত্তিক যোগাযোগ, কেনাকাটা, ব্যাবসা-বাণিজ্য এবং আরো অনেক ক্ষেত্রে। এসব যোগাযোগের কারণে অসংখ অসংরক্ষিত পাঠ্য উপাত্ত ওয়ার্ল্ড ওয়াইড ওয়েবে জমা হচ্ছে। এসব অসংরক্ষিত পাঠ্য উপাত্তের মধ্যে রয়েছে বিষাক্ত তথ্য যেমন, গুজব, মিথ্যা-তথ্য, এবং দেশবিদ্রোহি তথ্য। এসব বিষাক্ত তথ্যের উপর ভিত্তি করে, সাধারণ মানুষ তৈরী করতে পারে অপ্রত্যাশিত ঘটনা যা দেশ ও জন জীবনের ক্ষতি সাধন করবে। এসব অপ্রত্যাশিত এবং বিষাক্ত তথ্যের প্রচারকে নিয়ন্ত্রণ করার জন্য কৃত্রিম বুদ্ধিমত্তা ভিত্তিক তথ্য উপাত্তের শ্রেণীবিন্যাস অতীব জরুরি। বাংলা ভাষায় কৃত্রিম বুদ্ধিমত্তা ভিত্তিক তথ্য উপাত্তের শ্রেণীবিন্যাস পদ্ধতি বিকশিত করা খুব কঠিন কাজ, কারণ কৃত্রিম বুদ্ধিমত্তা ভিত্তিক শ্রেণীবিন্যাস পদ্ধতি বিকশিত করতে হলে এই মডেলকে এই ধরণের বাংলা তথ্য দিয়ে প্রথমে প্রশিক্ষণ দিতে হবে এবং প্রশিক্ষনের পরে আমরা যে মডেল পাবো তা দিয়ে পরবর্তীতে এসব বিষাক্ত তথ্যের শ্রেণীবিন্যাস করা যাবে। বাংলা ভাষায় এসব কৃত্রিম বুদ্ধিমত্তা ভিত্তিক পদ্ধতি বিকশিত করতে হলে প্রথমে ট্যাগ ভিত্তিক বাংলা উপাত্তের প্রয়োজন যা বাংলা ভাষায় খুবই সীমিত। তথ্য উপাত্তের সল্পতার জন্য বাংলায় এই পদ্ধতি বিকশিত করা এখনও পিছিয়ে রয়েছে। এসব প্রেক্ষাপটের বিবেচনায়, এই গবেষণায় আমরা কৃত্রিম বুদ্ধিমত্তা ভিত্তিক তথ্য উপাত্তের শ্রেণীবিন্যাস পদ্ধতি বিকশিত করবো এবং বিকশিত করতে বাংলা ট্যাগ ভিত্তিক তথ্য উপাত্ত (কর্পাস) সংগ্রহ করবো ।

প্রথম পর্যায়ে, বাংলায় কৃত্রিম বুদ্ধিমত্তা পদ্ধতি বিকশিত করতে দুটি কর্পাস বিকাশের কৌশল নিযুক্ত করা হয়েছে: ম্যানুয়াল ভিত্তিক এবং সক্রিয় ভিত্তিক পদ্ধতি, যার ফলে দুটি প্রধান কর্পাস তৈরি হয়েছে: এমবেডিং এবং টেক্সট শ্রেণীবিন্যাস কর্পোরা। এমবেডিং কর্পোরা এমবেডিং মডেল বিকশিত করার জন্য ব্যবহার করা হয়েছে। পাঁচটি এমবেডিং কর্পোরা তৈরি করা হয়েছে, যেমন: বিইসি ফর বেঙ্গলি অথোরশিপ ক্লাসিফিকেশন, ডাব্লিওইসি ফর বেঙ্গলি অথোরশিপ এট্রাইব্যুশন, কোভইসি ফর বেঙ্গলি কভিড টেক্সট আইডেন্টিফিকেশন, ইকোভই ফর ইংলিশ কভিড টেক্সট আইডেন্টিফিকেশন, এরাইসি ফর এরাবিক কভিড টেক্সট। টেক্সট শ্রেণীবিন্যাস কর্পোরা ট্রেনিং এবং টেস্টিং সেট এ বিভক্ত যেখানে সাতটি সুনির্দিষ্ট কর্পোরা রয়েছে, যেমন: বেঙ্গলি অথোরশিপ ক্লাসিফিকেশন কর্পাস (বিএসিসি-১৮), বেঙ্গলি টেক্সট কর্পাস (বিডিটিসিসি), এরাবিক টেক্সট ক্লাসিফিকেশন কর্পাস (এরাকোভ), বেঙ্গলি কভিড টেক্সট ক্লাসিফিকেশন কর্পাস (বিকোভসি), ইংলিশ কভিড কর্পাস (ইকোভসি), বেঙ্গলি টেক্সট ক্লাসিফিকেশন কর্পাস (বিটিসিসি১১), এবং বেঙ্গলি মাল্টি-মোডাল টেক্সট ক্লাসিফিকেশন কর্পাস (বিএমএমটিসি। এই কর্পোরা গুলি শ্রেণীবিন্যাস মডেলের পারফর্মেন্স এভালুয়েশনের জন্য অপরিহার্য।

দ্বিতীয় পর্বে, আমরা বাংলা এম্বেডিং কর্পোরার উপর ভিত্তি করে নন-প্রসঙ্গগত এম্বেডিং মডেল তৈরি করি এবং তাদের হাইপারপ্যারামিটারগুলিকে টিউনিং করি। এই এম্বেডিং গুলি তিনটি শ্রেণীতে বিভক্ত: ইন্ডিভিজুয়াল এম্বেডিং (গ্লোভ, ফাস্টটেক্সট, এবং ওয়ার্ড২ভেক), মেটা-এম্বেডিংস (এভিজি-এম, সিএটি-এম, সিএটি-এসবিডি-এম), এবং এটেনশন -বেসড মেটা-এম্বেডিংস (এপিএজিএফ এবং এপিসিজিএফ)। ইন্ট্রিন্সিক এভালুয়েশন শীর্ষ পাঁচটি পারফর্মিং মডেল সিলেক্ট করতে সাহায্য করে যা ডাউনস্ট্রিম টেক্সট শ্রেণিবিন্যাসের জন্য প্রয়োজন। আমরা প্রসঙ্গগত এম্বেডিংস এর মতো ট্রান্সফরমার - বেসড ল্যাংগুয়েজ মডেলস ব্যবহার করেছি ফাইন টিউনিং ও ফীচার এক্সট্রাকশন এর জন্। এই শীর্ষ পারফর্মিং মডেল গুলি পরবর্তী টেক্সট শ্রেণীবিন্যাস মডেল ডেভেলপমেন্ট এর ক্ষেত্রে গুরুত্বপূর্ণ ভূমিকা পালন করে।

তৃতীয় পর্যায়ে, টেক্সট শ্রেণীবিন্যাস মডেল তৈরি করতে এমবেডিং মডেলগুলিকে ব্যবহার করি এবং টেক্সট শ্রেণীবিন্যাসের মডেলকে বিভিন্ন পদ্ধতিতে প্রশিক্ষণ করানো হয়। এই মডেল গুলি তিনটি টেকনিক ব্যবহার করে তৈরী করি যেমন: স্ট্যাটিসটিকাল (এসভিএম, এনভি, আরএফ), ডিপ লার্নিং (সিএনএন, ভিডিসিএনএন, এলএসটিএম, জিআরইউ), ট্রান্সফরমার-বেসড মডেলস (বাংলাবার্ট ,মিউরিল, ডিবার্টএ-ভি৩, এক্সএমএল-রোবার্তা, ইন্ডিকবার্ট, এম-বার্ট, এরাবার্ট। হাইপার প্যারামিটার টিউনিং টেকনিক এই মডেল গুলিকে বেঙ্গলি টেক্সট শ্রেণীবিন্যাস এর ভিত্তিতে অপ্টিমাইজ করে। স্ট্যাটিসটিকাল সিগ্নিফিকেন্স টেস্ট দিয়ে শীর্ষ পারফর্মিং দুইটা মডেল বের করা হয় যা প্রস্তাবিত এটেনশন-বেসড আর্লি ফিউশন মডেল (এএফ ইউনেট) তৈরিতে ব্যবহৃত হয়। এএফইউনেট মডেল কে সূক্ষ্মভাবে ফাইন টিউনিং করা হয় বেঙ্গলি টেক্সট ক্লাসিফিকেশন কর্পোরার সাহায্যে যা চিত্তাকর্ষক একুরেসি অর্জন করে যেমন: ৯৬.৬০$\pm$0.২ বিটিসিসি১১ কর্পাস এর জন্য, ৮৫.৩৭$\pm$0.২ ওএসবিসি কর্পাস এর জন্য, ৯৭.৩৫$\pm$0.২ বার্ট কর্পাস এর জন্য, ৯৩.৭৪$\pm$0.২ ইন্ডিকএনএলপি কর্পাস এর জন্য, এবং ৯৬.৫১$\pm$0.২ প্রথম-আলো কর্পাস এর জন্য। এএফইউনেট এই পাঁচটি কর্পোরার ক্ষেত্রে পূর্ববর্তী স্টেট অফ দা আর্ট বেঙ্গলি টেক্সট শ্রেণীবিন্যাস মডেল এর চাইতে ভালো পারফর্ম করে যা 0.৫৪ হতে ৪.৪৯ পর্যন্ত একুরেসি ইম্প্রোভ করে।

Abstract

The exponential growth of unstructured textual data on the World Wide Web, particularly due to extensive online engagement and social media activities, poses a significant challenge in classifying and organizing this data efficiently. Resource-constrained languages like Bengali face specific hurdles, including a lack of annotated corpora, out-of-vocabulary word problems, inadequately tuned hyperparameters, and class distribution imbalances. To address these issues, this research introduces AFuNeT, i.e. Attention-based early fusion of transformer-based language models. This intelligent text classification system encompasses three crucial phases: corpora development, embedding model development, and text classification model development, offering a comprehensive solution to enhance text classification capabilities for such languages.

The first phase employs manual annotation and active learning techniques to develop two main corpus types: embedding and text classification corpora. Five specific embedding corpora are created for various tasks like Bengali document classification, authorship attribution, and Covid text identification. Text classification corpora are further divided into training and testing sets, comprising seven specific corpora, i.e., Bengali Authorship Classification Corpus (BACC-18), Bengali Text Classification Corpus (BDTCC), Arabic Covid Text Classification Corpus (AraCoV), Bengali Covid Text Classification Corpus (BCovC), English Covid Text Classification Corpus (ECoVC), Bengali Text Classification Corpus (BTCC11), and Bengali Multi-Modal Text Classification Corpus (BMMTC). These corpora are essential for text classification and model performance evaluation.

In the second phase, we've developed non-contextual embedding models using Bengali corpora and optimized their hyperparameters. These embeddings fall into three categories: individual (GloVe, FastText, Word2Vec), meta (AVG-M, CAT-M, CAT-SVD-M), and attention-based meta-embeddings (APeAGF, APeCGF). Top-performing models from intrinsic evaluations are selected for downstream tasks, with contextual embeddings like transformer-based models used for fine-tuning and feature extraction

In the third phase, we leverage our developed corpora and embedding models to create text classification models. We initially generate these models using three techniques: statistical (SVM, NB, RF), deep learning (CNN, VD-CNN, ACNN, LSTM, BiLSTM, GRU), and transformer-based language models (BanglaBERT, MuRIL, DeBERTa-V3, XML-RoBerTa, IndicBERT, DistilBERT, M-BERT, AraBERT). Hyperparameter tuning techniques optimize these models based on Bengali text classification corpora. The top-performing two models, as determined by statistical significance tests, are used to develop the proposed attention-based early fusion model, AFuNeT. AFuNeT undergoes meticulous fine-tuning using five Bengali text classification corpora, achieving impressive accuracies: 96.60±0.2 for BTCC11 corpus, 85.37±0.2 for OSBC corpus, 97.35±0.2 for BARD corpus, 93.74±0.2 for IndicNLP corpus, and 96.51±0.2 for ProthomAlo corpus. AFuNeT outperforms previous state-of-the-art Bengali text classification models on these five corpora, demonstrating accuracy improvements ranging from 0.54% to 4.49%.

Keywords: Natural Language Processing, Resource-Constrained Languages, Bangla Language Processing, Corpora Development, Embedding Corpora, Classification Corpora, Embedding Models, Intrinsic Evaluation, Meta-Embedding, Attention-based Embeddings, Classification Models, Hyperparameters tuning, Early-Fusion, Late-Fusion, Multi-head attention, Ablation Analysis, Statistical, Significance Test.

Contents

Acknowledgment i
সংক্ষিপ্তসার ii
Abstract iv
Acronyms xvi

1 Introduction **1**
1.1 Text Classification 2
1.1.1 Text Classification in Resource-constrained Languages . . 3
1.1.2 Text Classification in High-resource Languages 3
1.2 Motivations 4
1.3 Critical Challenges 5
1.4 Objectives 6
1.5 Contributions 7
1.5.1 Corpora Development 7
1.5.2 Embedding Models Development 9
1.5.3 Classification Models Development 10
1.6 Thesis Organizations 13
1.7 Summary 14

2 Background and Literature Reviews **15**
2.1 Text Classification: Related Terminologies 15
2.2 Types of Text Classification 19
2.3 Corpus Development Techniques 21
2.4 Text-based Embeddings 22
2.5 Text Classifications Techniques 24
2.6 Related Work 26
2.6.1 High-resource Text Classification 26
2.6.2 Resource-constrained Text Classification 29
2.7 Research Gaps 33
2.7.1 Research Questions and Answers 34
2.8 Problem Formulation 36
2.9 Summary 41

3 Corpora Development **42**
3.1 Corpora Development Techniques 42
3.1.1 Texts Source Selection 43
3.1.2 Crawler Development and Text Crawling 43
3.1.3 Text Preprocessing 45

3.2 Manual Annotation-based Corpora Development . . . 46
3.2.1 Annotation, Verification and Quality Measures for Text Data 46
3.2.2 Annotation, Verification and Quality Measures for Image-Text Data . . . 47
3.2.3 Corpora Statistics . . . 47
3.3 Active Learning-based Corpora Development . . . 55
3.3.1 Data Crawling . . . 56
3.3.2 Data Preprocessing . . . 57
3.3.3 Machine Annotation . . . 57
3.3.4 Post Annotations . . . 64
3.3.5 Verification . . . 65
3.4 Evaluation of the Corpora . . . 66
3.4.1 Qualitative Measures . . . 66
3.4.2 Quantitative Measures . . . 69
3.4.3 Intrinsic Evaluation Corpora . . . 71
3.4.4 Online Bengali Text Classification Corpora . . . 72
3.4.5 Online Bengali Authorship Attribution Corpora . . . 75
3.5 Summary . . . 77

4 Embedding Model Development **78**
4.1 Embedding Models . . . 78
4.2 Non-contextual Embeddings . . . 79
4.2.1 Embedding Models Hyperparameters Tuning . . . 80
4.2.2 Individual Embedding and Intrinsic Evaluations . . . 83
4.2.3 Meta-embedding and Intrinsic Evaluations . . . 87
4.2.4 Attention with Position-based Meta-embedding and Intrinsic Evaluation . . . 89
4.3 Contextual Embeddings . . . 91
4.3.1 Monolingual transformer-based language models . . . 92
4.3.2 Multilingual transformer-based language models . . . 92
4.4 Summary . . . 92

5 Development of Text Classification Models **94**
5.1 Text Classification Techniques . . . 95
5.2 Statistical Text Classifiers . . . 96
5.3 Deep Learning-based Text Classifiers . . . 97
5.3.1 Hyperparameters Tuning . . . 97
5.3.2 GloVe with CNN-based Model: Bengali Authorship Attribution . . . 98
5.3.3 GloVe with VDCNN-based Model: Bengali Text Classification . . . 103
5.3.4 Meta-Embedding with CNN-based Model: Bengali Text Classification . . . 107
5.3.5 Attention-based Meta-Embedding and ACNN-based Model: Bengali Covid Text Classification . . . 110
5.4 Transformer-based Text Classifiers . . . 114
5.4.1 Models Hyperparameters Tuning . . . 114

5.4.2 Fine-tuning for Arabic Covid Text Classification 115
5.4.3 Proposed Attention-based Early Fusion Network: AFuNeT 119
5.5 Summary 134

6 Experimental Results and Discussion 136
6.1 Experiment Settings and Environments 136
6.2 Evaluation Measures 137
6.2.1 Intrinsic Evaluators 137
6.2.2 Extrinsic Evaluators 138
6.3 Results 139
6.4 Performance of Embedding Models: Intrinsic Evaluation 139
6.4.1 Performance of Individual Embeddings 139
6.4.2 Performance of Meta-Embeddings 142
6.4.3 Performance of Attention-based Meta-Embeddings: 143
6.5 Performance of Text Classifiers: Extrinsic Evaluation 146
6.5.1 Performance of GloVe with CNN Model 146
6.5.2 Performance of GloVe with VDCNN Model 151
6.5.3 Performance of Meta-Embedding with CNN Model 157
6.5.4 Performance of Attention-based Meta-Embedding with ACNN Model 164
6.5.5 Extrinsic Performance of Transformer-based Models 170
6.5.6 Comparison of AuthorNet with Monolingual and Multilingual Approaches 196
6.5.7 Comparison of the Proposed AuthorNet with Existing Approaches 197
6.5.8 Extrinsic Evaluations: Text Classifiers Performance 198
6.5.9 Multimodal Text Classification Performance 200
6.5.10 Error Analysis 200
6.5.11 Error Analysis of Deep Learning-based Models 201
6.5.12 Quantitative Analysis 213
6.5.13 Qualitative Analysis 215
6.5.14 Impact of Text Classification in Multimodal Dataset . . . 216
6.5.15 Discussion 217
6.6 Summary 219

7 Conclusion and Future Works 220
7.1 Weaknesses of the Proposed Model 221
7.2 Possible Implications 222
7.3 Future Recommendations 223
7.4 Publications 223

List of Figures

1.1 A general overview of an intelligent text classification system . . . 3
1.2 Dissertation overview . . . 13

2.1 Abstract view of text classification for a resource-constrained language . . . 37

3.1 Abstract view of corpora development . . . 43
3.2 Abstract view of active learning-based semi-automated corpus development (ALSCD) framework . . . 56
3.3 Schematic representation of dictionary similarity-based annotation (DSA) scheme . . . 58
3.4 Impact of number of dictionary words on DSA Accuracy . . . 67
3.5 Average dictionary matching probability vs. second (T_{mn}) and first (T_{mx}) maximum word similarity scores . . . 68
3.6 Text annotation capability at the rank value of DSA algorithm . . . 68
3.7 Domain-wise distribution of crawled texts . . . 69
3.8 Variations of text size (in KB) with different word frequencies of *BEC* . . . 71
3.9 BARD, and IndicNLP corpora with category-wise text samples distribution . . . 75
3.10 BTCC1 and OSBC corpora with class-wise text samples distribution 75

4.1 Abstract view of Embedding Model Development . . . 79
4.2 Syntactic word pair (W1 = Transliteration: Parbata, Bengali to English Translation: Mountain & W2 = Transliteration: Parbatamālā, Bengali to English Translation: Mountains) feature distribution. AVG-M used the FastText & GloVe . . . 88
4.3 Semantic word pair (W1 = Transliteration: Abakasa, Bengali to English Translation: Vacation & W2 = Transliteration: Chuti, Bengali to English Translation: Holiday) feature distribution. AVG-M used the FastText & GloVe . . . 89

5.1 Abstract view of Text Classification Model Development: AFuNeT 95
5.2 Proposed authorship classification framework . . . 99
5.3 Proposed very deep convolution neural network-based text categorization . . . 104
5.4 Proposed VDCNN classifier architecture. A left green arrow indicates the backpropagation, and right black arrow denotes the forward propagation . . . 105

5.5 Overview of the proposed text classification model (AVG-M+CNN) 108
5.6 High-level View of CovTiNet 110
5.7 Abstract View of Proposed System: AraCovTexFinder 116
5.8 Abstract process of the AFuNet framework 119
5.9 Architecture of AFuNet: Text classifier for a low-resource language 124
5.10 Abstract view of the development of AuthorNet 128
5.11 Overview of text-to-feature extraction 129
5.12 Intelligent AuthorNet: Overview of attention-based early fusion of authorship attribution training and inferencing 131
5.13 Abstract view of multimodal text classification system 133

6.1 Performance of GloVe+CNN model on BACC-18, BAAD16 and LD. The error bar indicates the standard error. 149
6.2 Mean accuracy obtained by the human experts and VDCNN. Error bar indicates the standard deviation 156
6.3 Impact of position encoding (P_e) on embedding models for CTI task performance with CNN and ACNN 167
6.4 Impact of attention-based and positional embedding-based average feature fusions on CTI task performance with CNN and ACNN 168
6.5 Impact of Vocab-size and Language-specificity on Accuracy 174
6.6 Accuracy improvement between the best-performing eight baseline and fine-tuned models across five corpora 179
6.7 Summary of performance comparisons 184
6.8 Minimum performed hyperparameters performance of 8 transformer models with 4 corpora 189
6.9 Maximum performed hyperparameters performance of 8 transformer models with 4 corpora 189
6.10 Monolingual and Multilingual Best-2 Models Accuracy Comparison with the Proposed AuthorNet 197
6.11 Confusion Matrix of BACC-18 and corresponding indexes are referees as *Bankim Chandra Chattopadhyay (0)*, *Emon Jubayer (1)*, *Humayun Ahmed (2)*, *Humayun Azad (3)*, *Jibanananda Das (4)*, *Kazi Nazrul Islam (5)*, *Rabindranath Tagore (6)*, *Samaresh Majumdar (7)*, *Sharadindu Bandyopadhyay (8)*, *Sarat Chandra Chattopadhyay (9)*, *Satyajit Ray (10)*, *Shamsur Rahman (11)*, *Shirshendu Mukhopadhyay (12)*, *Sukumar Ray (13)*, *Sunil Gangopadhyay (14)*, *Swami Vivekananda (15)*, *Tareque Anu (16)* and *Taslima Nasrin (17)* 202
6.12 Confusion Matrix of BAAD16; corresponding indexes are referees as *Bankim Chandra Chattopadhyay (0)*, *Humayun Ahmed (1)*, *Manik Bandopadhyay (2)*, *Muhammed Zafar Iqbal (3)*, *Kazi Nazrul Islam (4*, *Nihar Ranjan Gupta (5)*, *Rabindranath Tagore (6)*, *Shirshendu Mukhopadhyay (7)*, *Samaresh Majumdar (8)*, *Shordindu Shekhor Chakma (9)*, *Sarat Chandra Chattopadhyay (10)*, *Satyajit Ray (11)*, *Sunil Gangopadhyay (12)*, *Tarasankar Bandyopadhyay (13)*, *Taslima Nasrin (14)* and *Zahir Raihan(15)* 203

6.13 Confusion Matrix of LD and corresponding indexes are referees as *Bankim Chandra Chattopadhyay* (0), *Rabindranath Tagore* (1) and *Sarat Chandra Chattopadhyay* (2) 204
6.14 Sample input/output with model performance 205
6.15 Confusion Matrix of BDTCC 205
6.16 Confusion matrix visualization for AVG-M+CNN model with BARD & BTCC11 corpora. BARD categories with index: economy(0), entertainment(1), international(2), sports(3) & state(4). BTCC11 categories with index: Accident(0), Climate-environment(1), Crime(2), Education(3), Entertainment(4), Health(5), Lifestyle(6), Politics(7), Science-Tech(8), Sports(9) & law-justice(10) 206
6.17 Confusion matrix visualization for AVG-M+CNN models using IndicNLP & Prothom-Alo corpora. IndicNLP categories with index: entertainment(0), international(1), kolkata(2), national(3), sports(4) & state(5). Prothom-Alo categories with index: economy(0), education(1), entertainment(2), international(3), life-style(4), opinion(5), sports(6), technology(7) 206
6.18 Few examples of failure text classification 207
6.19 Confusion matrix of the proposed model (CovTiNet) on of test samples 208
6.20 Actual and predicted test samples 209
6.21 Confusion matrix of AraCovTexFinder on test samples 210
6.22 Actual and predicted test samples 211
6.23 Few author text samples with attributed summary 212
6.24 confusion matrices of AFuNet model for the datasets: BARD, IndicNLP, and Prothom-Alo. BARD categories denoted by the indexes: economy(0), entertainment(1), international(2), sports(3) & state(4); IndicNLP categories represent with indexes: entertainment(0), international(1), Kolkata(2), national(3), sports(4) and state(5); Prothom-Alo categories indicates with indexes: economy(0), education(1), entertainment(2), international(3), lifestyle(4), opinion(5), sports(6), and technology(7) 213
6.25 Confusion matrices of AFuNet model concerning BTCC11 and OSBC corpora. The classes of BTCC11 dataset represents with various indexes: Accident(0), Climate-environment(1), Crime(2), Education(3), Entertainment(4), Health(5), Lifestyle(6), Politics(7), Science-Tech(8), Sports(9) and law-justice(10). The classes of the OSBC dataset are denoted with various indexes: Entertainment (0), Opinion (1), Crime (2), Economics (3), Sports (4), Art (5), Environment (6), International (7), Education (8), Politics (9), Science_tech (10), Accident (11) 214
6.26 Examples of text classification errors 215

List of Tables

3.1 Summary of corpora source text . . . 44
3.2 Bengali embedding corpus (*BEC*) statistics . . . 48
3.3 Key Statistics of Word Embedding Corpus (WEC) . . . 48
3.4 Statistics of *CovEC*) . . . 49
3.5 Statistical summary of ECoVE . . . 49
3.6 Arabic COVID-19 embedding corpus (*AraEC*) statistics . . . 50
3.7 BACC-18 Corpus Statistics . . . 51
3.8 Categorical distribution of (*BDTC*) . . . 51
3.9 *AraCoV* Corpus Statistics . . . 52
3.10 *BCovC* Corpus Statistics . . . 53
3.11 ECoVC Corpus Statistics . . . 54
3.12 BMMTC Corpus Statistics . . . 54
3.13 First & Second trail inter-annotator agreements with class wise kappa statistics . . . 66
3.14 Class-wise distribution of the classification corpus (*BTCC*11) . . . 70
3.15 Bengali embedding corpus (*BEC*) statistics . . . 70
3.16 Kappa scores for the intrinsic datasets. . . . 72
3.17 Statistical summary of text classification corpora . . . 74
3.18 Statistics of Authorship Classification Corpora (LD, BAAD6, BAAD16) 76

4.1 Initial and optimized hyperparameters of GloVe, fastText and Word2Vec for BEC and WEC . . . 84
4.2 Optimized hyperparameters for CovEC, ECoVE, and AraEC . . . 85

5.1 Text classification models development summary, ST: Statistical model, DL: Deep Learning-based Model, FN: Attention-based Early Fusion Networks, Y: Yes/Developed Model, N: Not/Not Developed Model . . . 135

6.1 Spearman ($\hat{\rho}$), Pearson ($\hat{r}$) and average cosine similarity (A_{CS}) statistics for 200 semantic, syntactic & relatedness word pairs. . . . 140
6.2 Semantic Syntactic and relatedness word analogy tasks performance summary . . . 140
6.3 Summary of the intrinsic evaluation results for best-performing embedding techniques based on Spearman ($\hat{\rho}$) and Pearson ($\hat{r}$) correlations . . . 141
6.4 Intrinsic performance of the best-performed embedding models . . . 141

6.5 Performance of the best-performed embedding models for analogy tasks regarding 200 semantic, 200 syntactic & 200 relatedness word-pairs. . . . 142
6.6 Spearman ($\hat{\rho}$), Pearson ($\hat{r}$) and average cosine similarity (A_{CS}) statistics for 200 semantic, syntactic & relatedness word pairs. GloVe & FastText (Skip-gram) features are used in AVG-M, CONC-M & CONC-SVD-M. CONC-M embedding dimension of 600 & others is of 300. . . . 143
6.7 Semantic Syntactic and relatedness word analogy tasks performance summary. GloVe & FastText (Skip-gram) features are used in AVG-M, CONC-M & CONC-SVD-M. CONC-M embedding dimension of 600 & others is of 300. . . . 143
6.8 Intrinsic performance of the best-performed attention-based meta-embedding models . . . 144
6.9 Performance of the best-performed attention-based meta-embedding models for analogy tasks regarding 200 semantic, 200 syntactic & 200 relatedness word-pairs. . . . 145
6.10 Macro-F1 (M_{ap}), weighted-F1 (WA_p) and Accuracy (A) measures on three datasets . . . 147
6.11 Statistical measures summary of GloVe+CNN with 250 ED on three datasets). . . . 148
6.12 Performance comparison of authorship attribution . . . 149
6.13 Performance of transformer models in authorship classification . . . 150
6.14 Bengali text classification task performance of26 best-performed models . . . 151
6.15 Statistical evaluation of document classifier . . . 153
6.16 Summary of previous Bengali document classification techniques concerning training size, testing size, and classes . . . 154
6.17 Summary of the performance comparison . . . 154
6.18 Performance comparison between human experts and proposed System . . . 156
6.19 Summary of CNN performance with initial, tuned & KerasTuner using corpus BTCC11 . . . 157
6.20 Test phase statistical performance of individual embeddings with different classifiers, Corpus, testing texts (Ts) . . . 158
6.21 Test phase statistical performance of meta-embedding with different classifiers, Corpus, testing texts (Ts) . . . 159
6.22 Summary of OOV impact on CNN accuracy . . . 161
6.23 Text classification test phase statistical performance of transformers-based and proposed models . . . 162
6.24 Comparison summary among proposed and previous research concerning accuracy (A), convergence time (T in hrs), corpus size (CS), and epoch numbers (E_p). . . . 165
6.25 CTI task performance of the proposed (CovTiNet) and baseline models. The MA and WA values are rounded up to two decimal points . . . 166

6.26 Comparison between the proposed and recent techniques in terms of A_c, TTH and $GMCG$ on $BCovC$ . . . 167
6.27 Performance of Arabic CovTI baseline methods where the bold indicates the score of top-4 models . . . 171
6.28 Performance of the fusion-based four best-performed models where the top model's score marked as bold . . . 171
6.29 Summary of hyperparameters and their impact on fine-tuned accuracy, B_S for batch size & M_{SL} for maximum sequence length (best combination marked as bold) . . . 173
6.30 Summary of OOV impact on accuracy where the top-4 scores marked as bold, CSM: character or subword-level match, OOV: out-of-vocabulary . . . 175
6.31 Performance of the Proposed system (AraCovTexFinder) and three best-performed where the top scored model marked as bold) . . . 175
6.32 Accuracy of baseline models on five corpora. The bold values indicate the best-performed models. Adding ±0.2 to all results is a way to express the potential variability or uncertainty in the reported performance metrics. . . . 177
6.33 Performance of the fine-tuned best-performed models with hyperparameters: focal loss (L^{Fo}) and categorical cross-entropy loss . . . 178
6.34 Performance of four best-performed late fusion-based techniques . 179
6.35 Performance of four best-performing attention-based early fusion models . . . 180
6.36 Summary of statistical significance test (t-value and p-value) of four models: $cat(MR_{r(1)}, MR_{r(3)})$, $cat(MR_{r(1)}, MR_{r(4)})$, and $cat(MR_{r(2)}, MR_{r(3)})$ models with five corpora compared to $cat(MR_{r(1)}, MR_{r(2)})$ 181
6.37 Summary of ablation analysis on $\Delta AFuNet$: Bold values highlight the maximum accuracy and key component configurations . . . 182
6.38 Impact of OOV words and tokenization operations on AFuNet accuracy across five corpora . . . 183
6.39 Performance summary of AFuNet . . . 184
6.40 Summary of comparison between the AFuNet and previous text classification models in Bengali . . . 186
6.41 Best-performed baseline methods Accuracy (A) and F1-score (F1) of four corpora, Bold values indicate the best-performed models . 187
6.42 Statistical performance summary of 8 transformer-based fine-tuned models with 4 corpora, Bold values indicate the best and second best-performed models . . . 191
6.43 Performance summary of attentional and non-attentional early fusion . . . 192
6.44 Statistical significance test t-value and p-value summary of transformer-based models with four corpora compared to AuthorNet, Bolded values indicate that the difference is statistically significant . . . 193
6.45 AuthorNet component-level ablation analysis summary . . . 194
6.46 Impact of OOV words and tokenization operation on AuthorNet accuracy . . . 196

6.47 Performance compression of proposed AuthorNet and exiting approaches 198
6.48 Summary of top-performing text classification across different corpora 199
6.49 Performance of Bengali multimodal text classification on BMMTC Corpus 200
6.50 Summary of multimodal text classification on BMMTC corpus . . 216

1 Sample inputs and output based on proposed AFuNeT model . . 239
2 Sample inputs and output based on proposed AFuNeT model . . 240
3 Sample inputs and output based on proposed AFuNeT model . . 241

Acronyms

NLP Natural Language Processing
ML Machine Learning
DL Deep Learning
TL Transformer-based Learning
LM Language Models
ITSC Intelligent Text Classification System
GloVe Global Vectors for Word Representation
TF-IDF Term Frequency - Inverse Document Frequency
OOV Out-of-Vocabulary
AFuNeT Attention-based Fusion of Transformers Network
BEC Bengali Embedding Corpus
WEC Word Embedding Corpus
CovEC Covid Embedding Corpus
ECoVE English Covid Embedding Corpus
AraEC Arabic Embedding Corpus
SOTA State-of-the-Art
S_s Semantic Word Similarity Dataset
S_y Syntactic Word Similarity Dataset
S_r Relatedness Word Similarity Dataset
A_t Analogy Tasks Dataset
AA Authorship Attribution
BACC-18 Bengali Authorship Classification corpus with 18 Authors
BDTCC Bengali Text Classification Corpus
AraCoV Arabic COVID-19 Text Classification Corpus
BCovC Bengali COVID-19 Text Classification Corpus
ECoVC English Covid Text Classification Corpus
BARD Bengali Article Classification Corpus
OSBC Open Source Bengali Corpus
IndicNLP Indian Bengali Text Classification Corpus
Prothom-Alo Online Bengali Newspaper Text Classification Corpus
LD Literature Dataset
BAAD16 Bengali Authorship Attribution Dataset with 16 Authors
BAAD6 Bengali Authorship Attribution Dataset with 6 Authors
BMMTC Bengali Multi Modal Text Classification Corpus
BTCC11 Bengali Text Classification Corpus

CNN Convolutional Neural Networks
ACNN Attention-based Convolutional Neural Networks
RNNs Recurrent Neural Networks
LSTM Long Short-Term Memory
BiLSTM Bidirectional Long Short-Term Memory
GRU Gated Recurrent Units
VDCNN Very Deep Convolutional Neural Networks
BERT Bidirectional Encoder Representations from Transformers
DeBERTa Decoding-enhanced BERT with Disentangled Attention
BPE Byte-Pair Encoding
NB Naive Bayes
LR Logistic Regression
SVM Support Vector Machines
LibSVM A Library for Support Vector Machines
LDA Latent Dirichlet Allocation
ANOVA Analysis of Variance
SMOTE Synthetic Minority Oversampling Technique
BoW Bag of Words
SGD Stochastic Gradient Descent
ALSCD Active Learning-based Semi-automated Corpus Development
GS Grid Search
RT Ray Tune
AVG-M Averaging-based Meta-Embedding
APeAGF Attention-based Meta-Embedding of Averaging of GloVe with FastText
H_j^I Initial Hyperparameters
H_j^T Tuned Hyperparameters
$\Psi_L(.,.)$ Function of TL-based Baseline Models Hyperparameters Tuning
$\Psi_D(.,.)$ Function of DL-based Baseline Models Hyperparameters Tuning
$\Psi_V(.,.)$ Function of Text Classification Models Evaluation
L^{Cs} Categorical-cross-entropy Loss
L^{Fo} Focal Loss
$\Phi_{pz}(.,.)$ Function of Late Fusion-based Evaluation
$\lambda_{WF}(.,.)$ Best-performing Model Selection Function
$\delta(.,.)$ Ablation Analysis Function

P_o Relative Observed Agreement
P_e Hypothetical Probability of Chance Agreement
K Kappa Coefficient
DSA Dictionary Similarity-based Annotation
$ORS(.)$ Optimal Rank Selection
ED Embedding Dimension
CW Contextual Windows
CBOW Continuous Bag of Words
ME Meta-Embedding
lr Learning Rate
H_m Semantic Human Judgement Similarity Score
H_y Syntactic Human Judgement Similarity Score
H_r Relatedness Human Judgement Similarity Score
H_{at} Analogy Tasks Human Judgement Similarity Score
CONC-M Concatenation-based Meta-Embedding
CONC-SVD-M Concatenation with Singular Value Decomposition-based Meta-Embedding
$\bigotimes$ Element-wise Multiplication
$\bigoplus$ Concatenation
FM Feature Matrix
B_m Best-performing Classifiation Models
$\binom{N}{M}$ Binomial Combinations of M taken from N
CLS Classification Features of Transformer-based Model
E_d Classification Model Embedding Dimension
Q Query Matrix of Attention Layer
K Key Matrix of Attention Layer
V Value Matrix of Attention Layer
$\overrightarrow{A_w}$ Feature Vector of Word A
$\overrightarrow{B_w}$ Feature Vector of Word B
C_{cs} Cosine Similarity Score
$\hat{\rho}$ Spearman Correlation
$\hat{r}$ Pearson Correlation
A Classification Model Accuracy
P Classification Model Precision
R Classification Model Recall

$F1 - score$ Averaging of Precision and Recall
MA Macro Average
WA Weighted Average
WA_a Weighted Average Precision
WA_r Weighted Average Recall
M_{ap} Macro-Average Precision
M_{ar} Macro-Average Recall
Fp Friedman Test Probability Value
Ft Friedman Test Variance Value
$p - value$ Welch's t-test Probability Value
$t - value$ Welch's t-test Variance Value

Chapter 1

Introduction

In the contemporary era, characterized by the Fourth Industrial Revolution, the globe is intricately interconnected through the Internet and mobile communication networks. This interconnectedness has given rise to a plethora of online digital media platforms, such as Facebook, Twitter, Instagram, YouTube, Tumblr, LinkedIn, Snapchat, and TikTok, collectively generating an enormous volume of textual data. For instance, Twitter alone witnesses the creation of 500 million tweets each day, underscoring the vast potential for valuable information extraction [1]. Despite the wealth of information embedded in this textual data, a significant drawback lies in its unstructured nature. This unstructured data, comprising documents, web files, business and medical reports, tweets, Facebook posts, and more, poses a formidable challenge for practical utilization by industries, businesses, governments, health services, and various public and private organizations. One promising avenue to unlock the potential of unstructured text is the application of machine learning techniques, specifically text classification. By employing machine learning models, this extensive and unorganized textual data can be categorized and organized. This classification facilitates effective management, aids businesses in decision-making, and provides valuable insights for policymakers.

In this chapter, we will provide a succinct overview of text classification, exploring its applications and significance in the context of the information-rich digital landscape. The focus will then shift towards a specialized domain – resource-constrained text classification. Motivated by the challenges posed by resource constraints in various environments, we will delve into the nuances of resource-constrained text classification tasks. This specialized form of text classification is driven by the need to operate efficiently in settings where computational resources, data availability, or both, are limited. Our exploration will outline the motivations behind emphasizing resource-constrained text classification. We will

elucidate the primary driving factors that propel the development of solutions tailored for scenarios where resources are scarce but the demand for effective text classification remains high. The chapter will also shed light on the significant challenges inherent in developing resource-constrained text classification systems. From model complexity to data scarcity, we will address the intricacies involved in creating robust and efficient solutions for constrained environments. In conclusion, we will highlight the critical contributions stemming from the endeavors in resource-constrained text classification system development. By summarizing the key insights and advancements in this field, we aim to underscore the importance of harnessing machine learning techniques to transform vast unstructured textual data into structured, actionable information in resource-limited settings.

1.1 Text Classification

Text classification (TC) is a fundamental research problem in Natural Language Processing (NLP) that involves assigning categories or class labels to various textual units, spanning from paragraphs and phrases to queries and entire documents. Recently, NLP research efforts have increasingly concentrated on tackling challenges in resource-constrained languages, including languages like Tamil, Hindi, Bengali, Kannada, and even Latin [2]. Bengali, for instance, stands out as the fifth most widely spoken native language worldwide and is a prominent example of a resource-constrained language within the NLP landscape [3]. It's worth noting that approximately 245 million people in Bangladesh and certain regions of India communicate using the Bengali language [4]. Over recent years, the proliferation of digital Bengali text content on the Internet, including news portals, blogs, websites, and more, has been remarkable, primarily due to the widespread use of electronic devices. This burgeoning content has generated an immense volume of unstructured data, presenting a substantial challenge in human manual organization, search, and manipulation. This manual effort is not only time-consuming but also costly. Therefore, developing an intelligent document categorization system is essential, capable of efficiently handling vast quantities of Bengali text data facilitating the swift and effective sorting, manipulation, and organization of documents. An intelligent text classification system overview is presented in Fig. 1.1.

In Figure 1.1, we observe that two distinct input types are utilized: pure text and images accompanied by text. These inputs are directed into an intelligent text classification system (ITSC). The ITSC operates through a sequence of three fundamental stages: (i) input transformation, (ii) feature extraction, and

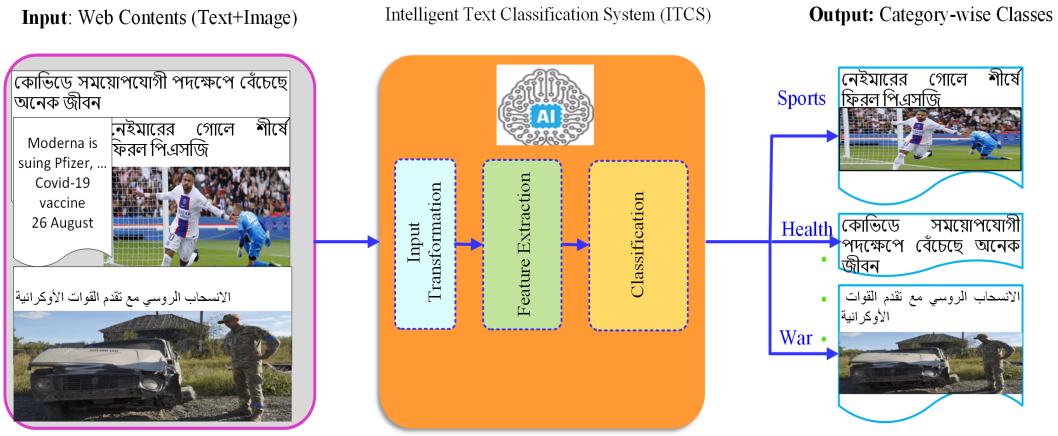


Figure 1.1: A general overview of an intelligent text classification system

(iii) classification. Ultimately, this classification phase yields the anticipated category. This overview encapsulates the fundamental essence of an intelligent text classification system.

1.1.1 Text Classification in Resource-constrained Languages

Text classification in resource-constrained language refers to the process of categorizing or labeling text data written in languages that face limitations in terms of available linguistic resources, such as annotated datasets, pre-trained embedding models, pre-trained classification models, lack of well-tuned hyperparameters, and natural language processing tools. Tamil, Hindi, Bengali, Kannada, and even Latin are the best examples of resource-constrained languages [5]. In this context, Bengali text classification aims to assign categories or labels to text documents, but it encounters difficulties due to the scarcity of language-specific resources.

1.1.2 Text Classification in High-resource Languages

High-resource language text classification refers to categorizing text data written in languages for which ample linguistic resources, such as large annotated corpora and pre-trained language models, are readily available [6]. In this context, high-resource languages typically include English, Spanish, and Chinese, which have extensive linguistic resources and well-established NLP tools and models. Researchers working on high-resource language text classification often have access to a wide range of pre-trained embeddings, robust language models, and sizable labeled datasets, allowing for the developing of sophisticated and highly accurate text classification systems. As a result, text classification in high-resource

languages has achieved advanced performance and sophistication, making it a well-studied and rapidly evolving field within natural language processing (NLP).

1.2 Motivations

In this research, resource-constrained text classification is primarily motivated by five distinct factors. Subsequent paragraphs will provide a detailed description of each of these motivating factors.

First Motivation The proliferation of the Internet and electronic devices has ushered in a substantial influx of digitized Bengali textual content onto the World Wide Web. These texts originate from diverse sources, including social media, online newspapers, blogs, medical services, product reviews, and various text-based communication platforms. A substantial portion of these textual interactions manifests in unstructured and disorganized formats. This situation poses considerable challenges, particularly as government bodies and security agencies grapple with disseminating fake news, toxic content, and anti-government sentiments propagated through text-based communication channels. The manual detection and organization of this vast volume of unstructured textual data by human experts are impractical due to the substantial time and financial investments required. Developing an intelligent Bengali text classification system is a pivotal solution to address these issues effectively. Such a system can play a crucial role in controlling the spread of fake news, accurately identifying toxic content, and providing an automated and efficient approach to sorting, organizing, and processing textual data. Therefore, the primary motivation behind this research endeavor is the pressing need for these applications.

Second Motivation In recent times, limited research has focused on Bengali text classification tasks. Much of the existing research predominantly concentrates on text-to-feature extraction and the generation of text classification models. Many studies employ statistical methods (such as N-grams and TF-IDF) and non-contextual feature extraction methods (such as GloVe, FastText, and Word2Vec) for text-to-feature extraction. However, these feature extraction approaches fall short of capturing context-aware semantic and syntactic information and do not adequately address the inclusion of features for Out-of-Vocabulary (OOV) words. These limitations underscore the second motivation for this research: developing a text classification system that seamlessly incorporates contextual knowledge.

Third Motivation Due to morphological variations (such as Sadhu-bhasha and Cholito-bhasha), hyperparameter tuning for high-resource text classification models cannot be directly applied to Bengali language models. Existing Bengali classification models predominantly focus on hyperparameter tuning for convolutional and sequential deep learning systems, neglecting the crucial aspect of hyperparameter tuning for transformer-based language models. This deficiency forms the third motivation for the development of a Bengali text classification system.

Fourth Motivation Transformer-based language models face challenges in extracting generalized embedding features for Bengali text classification tasks. These challenges arise from the limited vocabularies within language models and the dearth of transferable domain knowledge in pre-trained models. Most language models are trained across various languages, with Bengali text representation in their training datasets being notably limited compared to other languages. Consequently, transformer models trained on the same datasets exhibit inconsistent performances. Addressing this challenge represents the fourth motivation behind this research.

Fifth Motivation The issue of class imbalance in text classification corpora poses a significant and formidable challenge, particularly in the context of low-resource text classification tasks. This class imbalance significantly impedes classification performance, and remarkably, none of the existing Bengali text classification approaches have adequately addressed this issue. This gap in the literature has motivated us to develop a Bengali text classification system that explicitly tackles the class imbalance problem, serving as the fifth motivation for this research.

1.3 Critical Challenges

Low-resource Bengali text classification presents several challenges due to the scarcity of labeled data and resources. Some of the key challenges include:

- **Limited Labeled Data:** Bengali, particularly in specialized domains, often lacks an adequate amount of labeled text data for training and evaluating classification models.
- **Language Variability:** Bengali dialects, regional variations (i.e., Sadhu-bhasa and Chalit-bhasa), and script variations pose challenges in creating

a consistent and accurate text classification model.

- **Lack of Standard Datasets:** Unlike languages with more resources, Bengali may not have well-established and widely recognized benchmark datasets for various classification tasks.
- **Imbalanced Data:** In the Bengali text dataset, class imbalances can be pronounced, leading to biased models and challenges in handling minority classes.
- **Shortage of Well Tuned Hyperparameters:** In the domain of Bengali text processing, there is a lack of finely tuned hyperparameters for embedding and classification models.
- **Lack of Pre-trained Models:** Owing to the limited linguistic resources, highly effective embeddings and classification models for Bengali language processing have not yet been developed.
- **Out-of-Vocabulary Words:** Bengali text classification models may struggle with handling out-of-vocabulary words or phrases, particularly in specialized or evolving domains.
- **Domain Adaptation:** Adapting models trained in resource-rich languages to low-resource Bengali domains can be challenging due to domain-specific terminology.
- **Cross-Domain Transfer:** The lack of data in specific domains may necessitate the transfer of knowledge from related domains, which can lead to reduced performance.
- **Limited Language Technologies:** Low-resource languages like Bengali may have limited access to natural language processing tools, such as part-of-speech taggers, sentiment analyzers, and named entity recognition systems.

Addressing these challenges typically involves creative solutions such as data augmentation, active learning, transfer learning from resource-rich languages, and community efforts to create and share Bengali language resources and tools.

1.4 Objectives

As the ultimate objective of this research, we address the identified limitations within the field of Bengali text classification, encompassing challenges like corpus

development, non-contextual feature extraction, OOV word handling, insufficient hyperparameters tuning, limitations in language model feature generalization, and the prevalent issue of class imbalance in corpora. In response to these challenges, we have devised an innovative solution, an intelligent and context-aware Bengali text classification system known as AFuNeT. AFuNeT incorporates an active learning-based corpus development system, which minimizes the corpus development issues. Transformer-based language model to effectively mitigate issues related to non-contextual features and OOV words. Additionally, the application of attention-based early fusion serves to rectify shortcomings concerning feature generalization. To further enhance classification performance, AFuNeT fine-tunes hyperparameters through grid search and addresses class imbalance issues using a focus Focal Loss (FL) optimization technique. By successfully addressing these intricate challenges, AFuNeT emerges as a groundbreaking contribution to the field of Bengali text classification. This achievement not only advances the state-of-the-art but also opens up promising avenues for future research endeavors in this domain.

1.5 Contributions

In this section, we address the challenges posed by resource-constrained text classification problems through four significant contributions. These contributions are outlined in the following subsections.

1.5.1 Corpora Development

The *first* contribution is the corpora development within the thesis focuses on developing corpora foundational aspects of resource-constrained text classification systems. To address this challenge, the research identified three sub-problems, namely:

(i) Automatic Data Scraper Development: To tackle this issue, we developed a Python-based data scraping tool capable of autonomously extracting text and image data from the web while adhering to the 'robots.txt' policy.

(ii) Data Pre-processing and Annotation: Data pre-processing and annotation are pivotal tasks in the development of low-resource text classification systems. In response, we devised an automatic pre-processing tool to eliminate unwanted symbols and tokens. Additionally, we employed manual data annotation to label the pre-processed data.

(iii) Active Learning-Based Semi-Automated Corpus Development:

This sub-problem led to the creation of an active learning-based corpus development system. Through this system, we successfully mitigated the challenges posed by resource constraints. As a result, three distinct types of corpora were developed:

Embedding Corpora:

- BEC: Bengali Embedding Corpus (For text classification)
- WEC: Word Embedding Corpus (For authorship attribution)
- CovEC: Covid Embedding Corpus (For Bengali Covid text identification)
- ECoVE: English Covid Embedding Corpus (For English Covid text identification)
- AraEC: Arabic Embedding Corpus (For Arabic Covid text identification)

Intrinsic Evaluation Corpora:

- S_s: Semantic Word Similarity Dataset
- S_y: Syntactic Word Similarity Dataset
- S_r: Relatedness Word Similarity Dataset
- A_t: Analogy Tasks Dataset

Text Classification Corpora:

- BACC-18: Bengali Authorship Classification Corpus (For Bengali Authorship Attribution)
- BDTCC: Bengali Text Classification Corpus (For Bengali text classification)
- AraCoV: COVID-19 Text Classification Corpus (For Arabic Covid text classification)
- BCovC: Bengali Covid Text Classification Corpus (For Bengali Covid text classification)
- ECoVC: English Covid Text Classification Corpus (For English Covid text classification)
- BMMTC: Bengali Multi-Modal Text Classification Corpus (For Bengali Multi-Modal Classification)

- BTCC11: Bengali Text Classification Corpus (For Bengali text classification)

A comprehensive overview of the solutions to these three sub-problems, along with detailed statistics and characteristics of these corpora, is provided in Chapter 3.

1.5.2 Embedding Models Development

The development of embedding models represents the *second* significant contribution of this research. Given the scarcity of built-in Bengali embedding models, this study initially tuned the embedding model hyperparameters. In the text-to-feature extraction phase, each word is mapped to a real-valued vector, known as an embedding feature. Given the scarcity of low-resource embeddings, this study developed Bengali embedding models hyperparameters tuning methods using three distinct embedding methods: Word2Vec, FastText, and GloVe. However, the default hyperparameters designed for high-resource languages fail to adequately capture semantically and syntactically correlated features due to the presence of low-resource morphological variations. To address this challenge, this study introduced an automatic embedding hyperparameter tuning method. This method maximized the intrinsic performance of the embedding models by fine-tuning critical hyperparameters such as window size, embedding dimension, minimum frequency words, and learning rate. After tuning the hyperparameters, creation of three distinct types of embedding models: (i) Individual Embedding, (ii) Meta-Embedding, and (iii) Attention-based Meta-Embedding.

(i) Individual Embedding Models: In this category, three different embedding techniques Word2Vec [7], GloVe [8], and FastText [9] were employed to produce a variety of embedding models based on the Bengali embedding corpus. Each technique generated embedding models using different combinations of hyperparameters.

- Word2Vec, using neural embeddings and local contextual window information.
- GloVe, creating embedding features based on both local and global word counts, with window size and local window-based context playing crucial roles.
- FastText, generating embedding models based on character n-grams, providing a robust solution for handling Out-of-Vocabulary (OOV) issues.

Each of these techniques resulted in unique embedding models, each with its strengths and characteristics.

(ii) Meta-embedding Models: This category involves the combination of different individual embedding models to create meta-embedding models. These meta-embedding models leverage the strengths of individual embeddings and offer improved representations of semantically and syntactically correlated features. Two main approaches were employed for meta-embedding [10]:

- Concatenated Meta-embedding Models: These models concatenate the embeddings from various individual models, preserving their distinct information.
- Averaging-Based Meta-embedding Models: These models calculate the average of embeddings from different individual models, creating a unified representation.

(iii) Attention-based Meta-embedding Models: The attention-based meta-embedding is fused with the individual embedding features based on attention values. In this research, we also include adding word position with attention-based features. This operation improved the syntactic feature quality [11].

The meta-embedding models provide a comprehensive solution that capitalizes on the strengths of individual embeddings, enhancing the overall quality of semantic and syntactic feature representation. These embedding models play a critical role in enhancing the performance of text classification models, enabling more accurate and contextually rich representations of textual data. The comprehensive details of the development of embedding models can be found in Chapter 4, where a thorough exploration of the techniques, processes, and outcomes related to creating these models is provided.

1.5.3 Classification Models Development

As part of the *third* major contribution, this research empirically employed the three major text classification approaches, i.e., (i) Statistical, (ii) Deep Learning-based, and (iii) Transformer-based. Due to the linguistics variations, hyperparameters of high-resource languages are not well performed. As a result, initially, we tuned the classification model's hyperparameters. The significance of hyperparameter tuning for text classification models becomes particularly pronounced, especially in low-resource languages. This research took on the pivotal task of finely adjusting the hyperparameters for a range of deep learning-based models,

including Convolutional Neural Networks (CNN) [12], Long Short-Term Memory networks (LSTM) [13], Bidirectional LSTMs (BiLSTM) [2], Gated Recurrent Units (GRU) [14], and Very Deep Convolutional Neural Networks (VDCNN) [15]. Furthermore, we meticulously fine-tuned the hyperparameters for transformer-based language models, encompassing BERT-based [16], ELECTRA-based [17], and DeBERTa-based models [18]. In the following, we described the shortly each of the classification models.

Statistical Approaches: Four statistical classification models are used in this research to investigate the Bengali text classification models. Hear is a brief description of statistical models:

- Naive Bayes: A probabilistic model based on Bayes' theorem, known for its simplicity and effectiveness, especially for text classification tasks like spam detection [19].
- Logistic Regression: A regression model used for binary classification tasks, where it estimates the probability of a sample belonging to a particular class [20].
- Support Vector Machines (SVM): A supervised machine learning algorithm that finds a hyperplane that best separates data points into different classes [21].
- Decision Trees and Random Forests: Tree-based models that recursively split data based on the most significant features, making them interpretable and useful for text classification [22].

Deep Learning Approaches: In this research, we have conducted empirical evaluations on two variations of Convolutional Neural Networks (CNNs) and four variations of sequential deep networks. The details of each of these variations are presented below:

- Convolutional Neural Networks (CNN): A 1D CNNs are used to extract features from sequences of words. The CNNs use 3 kernels of convolutional layers to capture local patterns and relationships within the Bengali text.
- Very Deep Convolutional Neural Networks (VDCNN): The VDCNNs are an extension of CNNs used for Bengali text classification. The deep architecture with multiple 1D and 2D convolutional layers followed by pooling and fully connected layers [2].

- Long Short-Term Memory Networks (LSTM): LSTMs are a type of recurrent neural network (RNN) designed to capture sequential dependencies in Bengali text. The LSTMs have 2 hidden memory cells that can remember information over long sequences.
- Bidirectional LSTMs (BiLSTM): BiLSTMs enhance traditional LSTMs by processing Bengali text in both forward and backward directions. The 2 hidden layers of BiLSTMs are used in this research.
- Gated Recurrent Units (GRU): The GRUs are another type of recurrent neural network similar to LSTMs but with a simplified architecture. GRUs are effective for Bengali text classification tasks and are particularly useful when resource constraints are a concern

Transformer-based Approaches: This study conducted empirical fine-tuning of eight transformer-based language models using the Bengali text classification corpora. Below, we provide a short description of each of these models:

- BanglaBERT: This monolingual ELECTRA-based language model is pre-trained specifically for the Bengali language. B-ELECTRA: B-ELECTRA is an ELECTRA-based monolingual Bengali language model, pre-trained using a Bengali corpus.
- M-BERT: M-BERT is a multilingual language model that has been trained on a corpus containing a limited amount of Bengali text [16].
- MuRIL: This Indian languages-based language model includes Bengali and utilizes BERT as its backbone network [23].
- DeBERTa-V3: DeBERTa-V3 is a multilingual pre-trained language model with support for fine-tuning on Bengali-specific tasks [18].
- XLM-RoBERTa: XLM-RoBERTa is a multilingual optimized language model that builds upon the BERT architecture with optimizations for multilingual use [24].
- IndicBERT: IndicBERT is a language model designed to support Indian languages, including Bengali. It involves reducing some layers from the original BERT model during training [25].
- Distil-BERT: Distil-BERT shares a similar structure with the multilingual BERT model but differs in network architecture. It reduces some layers from the BERT model and is trained using the same corpus as used in the BERT model [26].

These transformer-based language models play a pivotal role in addressing the unique challenges of Bengali text classification, each offering its own set of advantages and characteristics tailored to the task at hand.

Following a thorough empirical analysis, this research has introduced a pioneering Bengali text classification system that leverages the early fusion of monolingual and multilingual transformer-based language models. This innovative system, aptly named **AFuNeT**, has demonstrated state-of-the-art (SOTA) performance across various Bengali text classification corpora. Comprehensive insights into the development of the Bengali classification model, AFuNeT, can be found in Chapter 6, where the system's architecture and performance are detailed.

1.6 Thesis Organizations

The thesis structure is illustrated in Figure 1.2. Chapter 1 serves as the introduction, offering insights into the foundational aspects of resource-constrained text classification, including motivating factors, challenges, objectives, and significant contributions.

Chapter 2 provides an in-depth background and literature review on text classification. This includes relevant terminologies, text classification technologies, related studies, research gaps, research questions, and problem formulation.

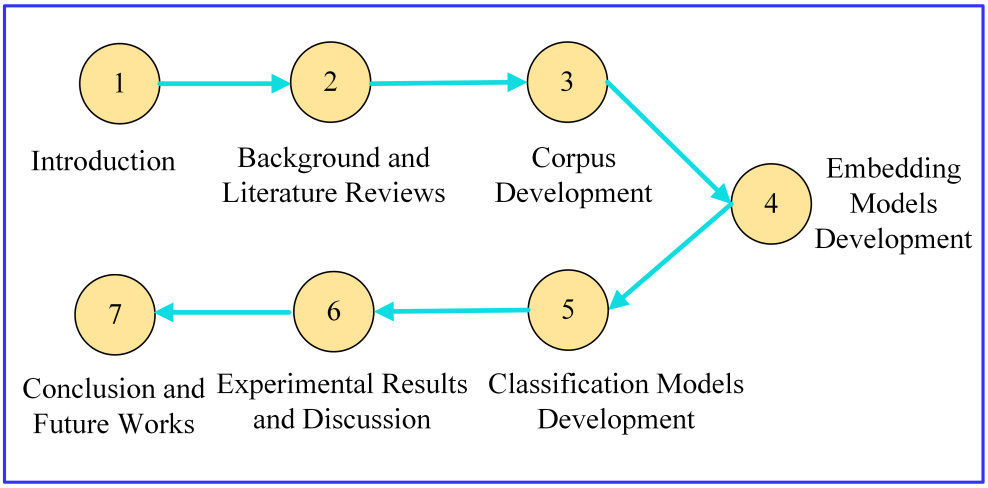


Figure 1.2: Dissertation overview

In Chapter 3, the focus shifts to corpora development, encompassing the creation of two primary types: (1) embedding corpora and (2) text classification corpora. The embedding corpora serve the purpose of extracting features during the text-to-features extraction phases, while the classification corpora are instrumental in developing classification models and verifying their performance.

In Chapter 4, the progress in embedding models is presented. This chapter introduces two categories of embedding models: (1) non-contextual and (2) contextual. The generation, evaluation, and selection of the best-performing embedding model for the non-contextual category will be outlined in this chapter. In contrast, contextual embeddings are employed as pre-trained models during the fine-tuning phase for text classification model generation purposes.

In Chapter 5, the attention turns to the depiction of text classification models. Three primary types of text classification techniques are employed, namely, (1) statistical, (2) deep learning, and (3) transformer-based learning, each utilizing various fusion methods.

In Chapter 6, the focus is on presenting experimental results and engaging in detailed discussions. This chapter provides an overview of the experimental environment, evaluation metrics, and the outcomes of our experiments. The results are organized into two main phases: (i) the findings related to embedding models and (ii) the performance of text classification models. We evaluate embedding models through intrinsic assessments and text classification models through extrinsic evaluations.

Chapter 7 functions as the conclusion, summarizing the essential findings and delineating future research directions. In this concluding chapter, we delve into the potential implications of our proposed research in resource-constrained text classification, offering insights into future research avenues to develop low-resource text classification systems.

1.7 Summary

This chapter provided an overview of text classification, emphasized the significance of addressing resource-constrained text classification challenges, and identified the specific motivations, challenges, objectives, and contributions that drive our research in developing a sophisticated Bengali text classification system. The subsequent chapters delve deeper into the background and literature reviews, corpora development, embedding models development, text classification models development, experimental results, and conclusion, ultimately demonstrating the novel contributions and advancements achieved in resource-constrained Bengali text classification.

Chapter 2

Background and Literature Reviews

Text classification is a prominent research challenge in natural language processing. Comprehensive background and literature reviews of text classification systems offer valuable insights into existing methodologies and challenges, providing a solid foundation for developing innovative approaches in text classification research. In this chapter, we commence by elucidating key terminologies associated with text classification tasks and subsequently delineating various types of text classification within the domain of natural language processing. Our focus extends to the intricacies of dataset development, embedding model creation, and text classification model construction. A comprehensive analysis of prior text classification research is undertaken, dissecting it within the contexts of both high-resource and resource-constrained scenarios. Within these scenarios, we investigate existing literature to discern research gaps, formulate pertinent research questions, and propose potential solutions. Ultimately, we culminate this chapter by presenting a well-defined problem statement that outlines the development of a *resource-constrained intelligent* text classification system.

2.1 Text Classification: Related Terminologies

In the following, we will explain a range of key text classification-related terminologies, providing definitions and explanations to ensure a clear understanding of the fundamental concepts in the field of text classification:

- **Text Classification:** The task of assigning predefined categories or labels to textual data based on its content [27].
- **Resource-constrained Languages:** Resource-constrained languages are

languages that lack adequate linguistic resources, such as annotated datasets, pre-trained models, and NLP tools, making text classification in these languages more challenging [2].

- **High-resource Languages:** High-resource languages have abundant linguistic resources, including large annotated corpora, pre-trained language models, and well-established NLP tools, which facilitate the development of sophisticated text classification systems [5].
- **Robots.txt:** The "robots.txt" policy simply means that the web crawler adheres to the rules and directives set in the "robots.txt" file when retrieving data from websites, ensuring that it doesn't access or crawl areas of the website that should be off-limits [28].
- **Active Learning:** A machine learning technique where the model selects the most informative data points to label and learn from, reducing the need for extensive labeled data [29].
- **Corpora** In the context of text classification and natural language processing, "corpora" is the plural form of "corpus." A corpus refers to a collection of written texts or documents used for linguistic analysis, research, or training natural language processing models [30].
- **Preprocessing:** The cleaning and transformation of text data, including tasks like tokenization, stemming, and stop-word removal.
- **Tokenization:** The process of dividing text into individual words or tokens.
- **Stop Words:** Common words that are often removed from text data as they don't carry significant meaning.
- **Bag of Words (BoW):** A text representation method that counts the frequency of words in a document without considering their order.
- **Term Frequency-Inverse Document Frequency (TF-IDF):** A method that measures the importance of a word in a document relative to its importance across the entire corpus [31].
- **Word Embeddings:** Continuous vector representations of words learned from large text corpora, used to convert text data into numerical form[8].

- **Contextual Embeddings:** Word embeddings that consider the context in which a word appears in a sentence or document, providing more nuanced representations [16].
- **Feature Extraction:** The process of transforming raw text data into a numerical format that can be used for machine learning algorithms [17].
- **Downstream Tasks:** Text classification tasks that build upon or use the output of earlier NLP tasks, often for more specific purposes such as authorship attribution, sentiment analysis, named entity recognition, or question answering [32].
- **Classifier:** A machine learning or deep learning model used to predict the category or label of text data [33].
- **Neural Networks:** Deep learning models used for text classification, including architectures like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) [34].
- **Pre-trained Models:** Models that have been trained on large text datasets and can be fine-tuned for specific text classification tasks [23].
- **Hyperparameters:** Parameters that are not learned by the model but are set prior to training, affecting the model's performance [35].
- **Transfer Learning:** A method where a pre-trained model is adapted for a specific text classification task, leveraging knowledge from a broader domain [36].
- **Ensemble Methods:** Combining multiple text classification models to improve overall performance, often using methods like bagging and boosting [37].
- **Unstructured Data:** Data that is not organized or formatted in a way that can be easily analyzed, common in the form of text documents.
- **Out-of-Distribution:** Refers to text data that is significantly different from the training data and may cause a text classification model to perform poorly or make incorrect predictions [38].
- **Label Noise:** Errors or inaccuracies in the labels of training data, which can negatively impact the performance of text classification models.

- **Multimodal Classification:** The task of classifying text data in conjunction with other types of data, such as images or audio, requiring fusion techniques to combine information from multiple modalities [39].
- **Fusion:** The process of combining information from different sources or models to improve the overall performance of a text classification system. This can include feature fusion, model fusion, or data fusion [40].
- **Early Fusion:** Early Fusion, or Feature-Level Fusion, integrates data from different sources or modalities at an initial stage, creating a unified representation. It is valuable when there's a strong correlation between data sources, as it captures cross-modal relationships. For example, in a system handling text and images, raw image data is combined with text features early in the process.
- **Late Fusion:** Late Fusion, or Decision-Level Fusion, combines information from different sources or modalities after individual processing. Each modality is independently processed, and their outputs are fused to make a final decision or inference. It is used when modalities offer complementary information for a more informed decision.
- **Statistical Significance Test:** In the context of text classification, a statistical significance test can be applied to evaluate the significance of differences in the performance of various classification models or techniques. For example, Welch's t-test, ANOVA test and Friedman test [41].
- **Intrinsic Evaluations:** Assessments of a text classification model's performance that are conducted within the context of the model itself, often involving metrics like word similarity, word analogy, or syntactic similarity, to evaluate the quality of word embeddings [42].
- **Extrinsic Evaluations:** Extrinsic evaluation is an assessment method in natural language processing and machine learning that evaluates a model based on its performance in real-world tasks or applications, as opposed to assessing its intermediate capabilities. Text classification involves testing the model's ability to categorize text documents for practical applications, such as organizing news articles. This evaluation measures the model's real-world utility and practical effectiveness [42].
- **Overfitting:** When a text classification model performs well on training data but poorly on unseen data because it has learned the training data too specifically [43].

- **Underfitting:** When a text classification model is too simple to capture the complexities of the data, resulting in poor performance [43].
- **Feature Engineering:** The process of creating new features or representations from the existing text data to improve classification accuracy.
- **Model Evaluation:** Assessing the performance of a text classification model using metrics like accuracy, precision, recall, and F1 score.
- **Cross-Validation:** A technique to assess a model's performance by splitting the data into multiple subsets for training and testing.
- **Zero-Shot Learning:** A text classification scenario where the model can classify text it has never seen during training, relying on its understanding of the data distribution [44].

2.2 Types of Text Classification

Text classification, also known as text categorization, is a genre within the field of natural language processing (NLP) and machine learning that focuses on automatically assigning predefined categories or labels to text data based on its content. This genre encompasses various specific tasks and applications, each with its characteristics and objectives. Here are some common genres or genres of text classification.

Document Classification Document classification is a broad category that involves categorizing entire documents, such as articles, emails, reports, or web pages, into predefined classes or categories. It is commonly used for tasks like spam email detection and news article categorization [45].

Sentiment Analysis Sentiment analysis, also known as opinion mining, is a specific genre of text classification that focuses on determining the sentiment or emotional tone expressed in text. It typically categorizes text as positive, negative, neutral, or sometimes more fine-grained sentiment categories [46].

Emotion Classification Emotion classification categorizes text based on the emotions expressed, which can be used in applications like customer feedback analysis and mental health monitoring [47].

Authorship Attribution Authorship attribution is the task of identifying the author of a text based on their writing style. It is often used in forensic linguistics and literary analysis [48].

Toxicity Detection Toxicity detection identifies toxic or offensive content in text, which is essential for maintaining safe online communities and preventing harassment [49].

Medical Text Classification In the healthcare domain, text classification is used for tasks like disease diagnosis, medical document categorization, and sentiment analysis of patient reviews [50].

Topic Classification Topic classification aims to categorize text documents into specific topics or subjects. It is widely used in news aggregation, content recommendation systems, and organizing large collections of textual data [51].

Intent Recognition Intent recognition is essential in chatbots and virtual assistants. It involves classifying user input text to understand their intent, enabling the system to provide appropriate responses or take relevant actions [52].

Language Identification Language identification is a genre of text classification that involves determining the language of a given text document, even when it contains multiple languages or code-switching [53].

Document Clustering Document clustering, although not strictly classified, groups similar documents together based on their content, allowing for the automatic discovery of document collections' underlying structure [54].

Named Entity Recognition (NER) Named entity recognition is a specialized form of text classification that identifies and classifies entities within the text, such as names of people, places, organizations, and dates [55].

Gender Classification Gender classification involves determining the gender of an author based on their written texts. This can be applied in various contexts, including social media analysis and customer profiling [56].

2.3 Corpus Development Techniques

Dataset development is a critical aspect of natural language processing (NLP) and machine learning, particularly for text classification tasks. Creating high-quality datasets is essential for training, validating, and testing machine learning models, and it involves various technologies and methodologies. Here's an overview of dataset development technology:

Data Collection: Data collection is the initial step in dataset development. Technologies for data collection can vary depending on the type of text classification task. Some common methods include web scraping, data crawling, and data acquisition through APIs. Web scraping tools like Beautiful Soup and Scrapy help extract text data from websites [57]. APIs allow access to specific data sources, like social media platforms or news websites.

Data Annotation: Annotated data is crucial for supervised learning in text classification. Data annotation involves labeling each text sample with its corresponding class or category. Crowd-sourcing platforms like Amazon Mechanical Turk and specialized annotation tools like Prodigy and Labelbox facilitate data annotation. Natural language processing tools like spaCy can also be used for entity recognition and tagging [58].

Data Preprocessing: Before adding data to a dataset, preprocessing is essential. This step involves cleaning and structuring the data, which may include tasks like removing irrelevant characters, handling missing values, and tokenizing text into words or phrases. NLP libraries such as NLTK, spaCy, and regular expressions are commonly used for data preprocessing [59].

Data Augmentation: In some cases, data augmentation technologies are employed to increase the size of the dataset. Data augmentation methods involve generating new data samples from existing ones by applying transformations such as synonym replacement, paraphrasing, or text translation. Tools like NLPAug and TextAttack can be useful for data augmentation.

Balancing Class Distribution: Ensuring a balanced class distribution is essential, especially when dealing with imbalanced datasets. Technologies for balancing class distribution include oversampling (duplicating samples from minority classes), undersampling (reducing samples from majority classes), and generating synthetic samples using techniques like SMOTE (Synthetic Minority Oversampling Technique) [60].

Data Storage: Storing datasets efficiently is crucial. Various database management systems (DBMS) like PostgreSQL, MongoDB, and cloud-based solutions like Amazon S3 and Google Cloud Storage can be used to store and manage large

text datasets. These technologies provide scalability and easy access to data.

Version Control: To keep track of dataset changes and updates, version control systems like Git can be used. This is particularly important when multiple contributors are involved in dataset development. Git helps in maintaining a history of changes and facilitates collaboration.

Privacy and Ethics: Dataset development must consider privacy and ethical concerns. Technologies for anonymization and data protection, like differential privacy and data masking, should be employed to ensure that sensitive information is not exposed.

Quality Assurance: Quality control and assurance technologies are essential to identify and rectify errors in the dataset. Manual review, automated checks, and validation scripts can be used to ensure data quality.

Data Splitting: Splitting the dataset into training, validation, and testing subsets is crucial for model development and evaluation. Technologies for data splitting include scripting and functions in data science libraries like scikit-learn.

In summary, dataset development involves a combination of technologies and methodologies, ranging from data collection and annotation to preprocessing, augmentation, storage, and quality assurance. Creating well-structured and representative datasets is foundational for successful text classification tasks and other machine-learning applications.

2.4 Text-based Embeddings

Embedding models are crucial in natural language processing tasks, including text classification. These models transform words or text into numerical representations, making it possible for machine learning algorithms to process and understand language. There are two types of embedding models in the NLP domain such as (i) Non-contextual Embedding and (ii) Contextual Embedding. Here's an overview of the technology involved in embedding model development:

Word2Vec: Developed by Google, Word2Vec creates word embeddings by training a shallow neural network on large text corpora. The Gensim library is commonly used to implement Word2Vec models [7].

GloVe (Global Vectors for Word Representation): GloVe is another popular method for creating word embeddings based on global word co-occurrence statistics. Pre-trained GloVe embeddings are available for various languages and can be integrated into NLP models [8].

FastText: Developed by Facebook, FastText is an extension of Word2Vec that considers subword information. It's particularly useful for handling out-of-

vocabulary words. Facebook's FastText library is available for training FastText embeddings [9].

Word2Vec, GloVe, and FastText Pre-trained Models: Pre-trained embeddings are readily available for various languages and can be directly used in NLP projects. Python libraries like Gensim and spaCy support these embeddings.

Pre-trained Models: Pre-trained embedding models are trained on large text corpora and can be fine-tuned for specific NLP tasks. Some of the widely used pre-trained embedding models and technologies include:

BERT (Bidirectional Encoder Representations from Transformers): BERT is a transformer-based model developed by Google. Pre-trained BERT models are available for multiple languages, and Hugging Face's Transformers library provides easy integration into NLP projects [16].

Word Embeddings from Transformers (e.g., RoBERTa, GPT-2): Transformers-based language models, like RoBERTa and GPT-2, can be fine-tuned for specific NLP tasks. Hugging Face's Transformers library offers pre-trained models and tools for fine-tuning [24].

Training Custom Embedding Models: In cases where pre-trained embeddings are not available or do not suit the task at hand, you can develop custom embeddings. Technologies for training custom embedding models include using deep learning frameworks like TensorFlow and PyTorch to build and train neural network models. You can train embeddings from scratch or use transfer learning techniques to leverage existing embeddings and fine-tune them for your specific task [58].

Evaluation of Embeddings: To assess the quality and effectiveness of embeddings, various evaluation datasets and metrics can be employed. Technologies for embedding evaluation include the use of benchmark datasets like WordSim-353 and evaluation metrics like cosine similarity or Spearman's rank correlation coefficient.

Hyperparameter Tuning: When developing custom embeddings, hyperparameter tuning is essential. Technologies for hyperparameter tuning include tools like Optuna and Hyperopt, which automate the search for optimal hyperparameters, such as the dimensionality of embeddings or training batch sizes [61].

Out-of-Vocabulary (OOV) Handling: Handling OOV words is a challenge in embedding models. Technologies like subword tokenization (e.g., Byte-Pair Encoding and SentencePiece) and character-level embeddings can help address OOV issues [62].

Transfer Learning: Transfer learning is a valuable technology when de-

veloping custom embeddings. It involves fine-tuning pre-trained embeddings on domain-specific or task-specific data. This can be achieved using deep learning frameworks like TensorFlow or PyTorch.

Visualization: Visualizing word embeddings is crucial for understanding their quality and relationships. Tools like t-SNE and TensorBoard are commonly used for embedding visualization.

Storage and Deployment: Storing and deploying embedding models efficiently is essential. Depending on the framework used, technologies like TensorFlow Serving, PyTorch Serving, and cloud-based solutions can be employed to serve embedding models in production.

In summary, embedding models development involves a range of technologies, from pre-trained embeddings and custom model training to evaluation, hyperparameter tuning, OOV handling, transfer learning, visualization, and deployment. Choosing the right embedding technology depends on the specific NLP task and available resources.

2.5 Text Classifications Techniques

Text classification model development technology involves the creation of machine learning or deep learning models that can automatically assign predefined categories or labels to textual data. These models play a crucial role in various natural language processing (NLP) tasks, including authorship attribution, document classification, topic categorization, and more. Here's an overview of the technology used in developing text classification models:

Feature Extraction: The process of converting raw text data into numerical features that machine learning algorithms can process. Traditional techniques like Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), and more recent word embeddings (Word2Vec, FastText, GloVe) are commonly used to represent text as feature vectors. The transformer-based language models can also represent more correlated and context-aware text features.

Supervised Learning: Text classification models are typically developed using supervised learning, where a labeled dataset is used for training. Each text document in the dataset is associated with one or more class labels, and the model learns to predict these labels based on the text content.

Algorithms and Models: Various algorithms and models can be used for text classification, including:

- **Naive Bayes:** A probabilistic model based on Bayes' theorem, often used

for its simplicity and effectiveness in text classification, especially for spam detection.

- **Logistic Regression:** A regression model used for binary classification tasks, estimating the probability of a sample belonging to a particular class.
- **Support Vector Machines (SVM):** A supervised machine learning algorithm that finds a hyperplane to separate data into different classes.
- **Decision Trees and Random Forests:** Tree-based models that recursively split data based on the most significant features, making them interpretable and useful for text classification.
- **Neural Networks:** Deep learning models like Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Long Short-Term Memory networks (LSTM), and more recently, transformer-based models like BERT and GPT are employed for text classification. These models can capture complex relationships and dependencies in text data.

Data Preprocessing: Preprocessing steps include tokenization, stop word removal, stemming or lemmatization, and handling special characters, numbers, and URLs. This ensures that the text data is in a suitable format for feature extraction.

Model Training: The model is trained on a labeled dataset, where it learns to make predictions based on the provided input features and labels. During training, optimization algorithms like stochastic gradient descent (SGD) are used to adjust model parameters for better performance.

Evaluation Metrics: Different evaluation metrics are used to assess the performance of text classification models. Common metrics include accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrices. The choice of metric depends on the specific problem and the importance of false positives or false negatives.

Hyperparameter Tuning: Fine-tuning hyperparameters is a crucial step in optimizing the performance of text classification models. This includes selecting the best learning rate, batch size, number of epochs, model architecture, and embedding dimensions, among others.

Cross-Validation: To avoid overfitting and assess model generalization, cross-validation techniques like k-fold cross-validation are employed. They divide the dataset into subsets for training and testing multiple times, providing a more robust evaluation of the model's performance.

Ensemble Methods: Ensemble methods like bagging and boosting can be used to combine the predictions of multiple text classification models to improve accuracy and robustness.

Transfer Learning: With the advent of transformer-based models, transfer learning has become a prevalent technique. Pre-trained language models like BERT are fine-tuned on specific text classification tasks, significantly reducing the need for large labeled datasets.

Regularization Techniques: Techniques like dropout and weight decay are applied to prevent overfitting and enhance the generalization of text classification models.

Interpretability: Model interpretability is becoming increasingly important. Techniques like attention mechanisms and gradient-based saliency maps are used to understand how the model arrives at its predictions.

Scalability: Developing models that can handle large volumes of text data efficiently is critical. Distributed computing and training on GPU or TPU clusters are employed to scale text classification models.

Model Deployment: Once a text classification model is developed, it needs to be deployed in a production environment, which involves creating APIs or integrating it with existing applications.

Text classification models development technology continues to evolve, with advancements in deep learning, transfer learning, and NLP, making it possible to create highly accurate and versatile models for a wide range of text classification tasks.

2.6 Related Work

In this section, we will provide a comprehensive overview of the findings and limitations of previous text classification tasks, drawing distinctions between high-resource and resource-constrained languages. By critically assessing the existing body of work, we aim to uncover the specific research gaps and constraints within these two distinct language domains, thereby laying the foundation for our innovative contributions to the field of Bengali text classification.

2.6.1 High-resource Text Classification

High-resource language text classification refers to the task of categorizing text data written in languages for which ample linguistic resources, such as large annotated corpora and pre-trained language models, are readily available. In this

context, high-resource languages typically include languages like English, Spanish, and Chinese, which have extensive linguistic resources and well-established NLP tools and models. In the text classification domain, Conneau et al. [63] developed a deep convolutional network (VDCN) text categorization system, achieving a remarkable accuracy of 98.71% on the DBPedia ontology classification dataset. They utilized 560,000 data points for training and 70,000 for testing across 14 categories. In contrast, [64] devised an unsupervised multi-tag text categorization system for the entertainment industry. Despite categorizing movie titles from internet databases into 100 categories, this system struggled to attain high accuracy due to limitations in feature extraction and classification techniques.

Moving forward, [6] introduced Deep Pyramid Convolutional Neural Networks (DPCNNs) for English text categorization. They achieved an accuracy of 97.36% on the Yelp dataset, which comprised 560,000 training documents, 38,000 testing documents, and 2 document classes. However, DPCNNs demonstrated a decrease in accuracy as the number of document classes increased, making it less suitable for tasks involving a higher number of classes. Additionally, character-level feature extraction and convolution network-based classifiers evolved over time, with an early application [65] achieving an impressive 98.69% test accuracy on the DBPedia ontology classification dataset with 14 distinct categories. However, this system faced the significant limitation of memory exhaustion, even when using two costly Tesla K40 GPUs.

Furthermore, [66] introduced a short text classification system based on recurrent and convolutional neural networks. This model was trained with 78,000 text documents, tested with 15,000, and validated with 16,000 documents from the Meeting Recorder Dialog Act (MRDA) Corpus [67], encompassing 5 categories. The system achieved an accuracy of 84.60%. Nonetheless, recurrent networks imposed substantial time requirements during both training and testing.

In addition, [68] proposed a semantic relation classification technique for document classification, featuring a feature extractor developed through negative sampling with global vector representation and a CNN-based classifier model. This system was trained with 8,000 text data points and tested with 2,717 labeled data points, achieving an accuracy of 85.60% across 10 semantic relations. Similarly, [46] developed a sentiment analysis system using the Word2Vec algorithm [7] for feature extraction and a Gated Recurrent Neural Network (GRNN) for sentiment classification. This system attained a 67.10% accuracy on the Yelp2014 dataset, which consisted of 1,125,457 training texts distributed across 5 categories. Additionally, [69] devised a text-based binary sentiment classification

system using a decision tree algorithm, achieving an accuracy of 91.81%. The system was trained with 7,000 Twitter texts expressing positive and negative sentiments and tested with 3,000 labeled Twitter texts. Notably, in sentiment classification systems, overall accuracy tended to decrease with an increasing number of sentiment/document classes. Moreover, [70] developed a Hierarchical Deep Learning (HDL)-based document classification system, extracting semantic features using the Global Vectors for Word Representation (GloVe) algorithm [8]. The classifier model was trained with 44,085 text documents and tested with 20,603 text documents across 7 categories, achieving an overall accuracy of 90.93%. Nevertheless, a combined approach of HDL and long short-term memory (LSTM) [45] with a generalized regression neural network (GRNN) led to increased training and testing times.

Notably, fuzzy clustering-based document categorization was proposed by [71], although this technique did not accurately determine the number of clusters required for unknown datasets. Meanwhile, [72] introduced an unsupervised clustering-based system for adaptive document categorization, employing approximately 20,000 newsgroups encompassing 20 different news topics. This system achieved an accuracy of 74.50%. Furthermore, [73] developed a document classification system in which the Word2Vec algorithm was used for feature extraction, and support vector machines (SVMs) [74] were employed for classification. This system was trained using the TREC news dataset [75], comprising 5,452 training documents and 500 test documents distributed across six categories, achieving an accuracy of 98.20%. Additionally, [76] proposed a document classification system where documents were represented using embedding models. Recurrent Attention Learning (RAL) with CNNs was employed for classification, and the system was trained using 33,388 text documents spanning 11 categories, achieving an accuracy of approximately 94.18%. Moreover, [77] developed a stemmer-based feature selection and decision tree-based classifier for Arabic text classification. This classifier was trained with 3,382 text documents across 6 categories, tested with 1,688 text documents, and achieved an accuracy of 79.00%. However, decision tree-based classifiers face challenges when trained with high-dimensional

Significant progress has been made in authorship classification research, particularly in high-resource languages like English and other European languages. Tweedle et al. [78] applied neural network techniques combined with stylometry for authorship classification in English texts [79]. Ruder et al. [80] introduced a character-level and multi-channel Convolutional Neural Network (CNN) approach for authorship attribution in English texts. Rocha et al. [81] explored various machine learning algorithms suitable for small sample sizes in authorship

classification. Koppel et al. [82] employed naive similarity-based techniques and achieved a precision of 93.20% for 1,000 authors in English text classification. In addition to English, authorship classification has extended to other languages. Kabala [83] developed a computational technique for authorship classification in a medieval Latin corpus. This technique involved the use of Bray–Curtis distance and logistic regression, resulting in an impressive accuracy of 99.86%.

In high-resource language text classification, various deep learning models such as transformer-based language models, VDCN, DPCNNs, and CNN-based classifiers have achieved impressive results, often with large training datasets. However, these models may face issues like memory exhaustion and reduced accuracy as the number of document classes increases. Additionally, some systems use semantic feature extraction methods like GloVe and Word2Vec for improved performance. These methods have achieved decent accuracy with smaller datasets but may encounter limitations in capturing semantic and syntactic features. Overall, there is a need for further research and tailored techniques in low-resource language text classification.

2.6.2 Resource-constrained Text Classification

Resource-constrained language text classification refers to the limitations regarding linguistic resources, annotated data, and pre-trained models. These languages, often considered low-resource languages, include languages like Bengali, Tamil, and many indigenous or less widely spoken languages. Text classification in resource-constrained languages presents unique challenges due to the scarcity of linguistic resources and domain-specific data. This research working in the Bengali text classification domain aims to develop text classification systems that can perform effectively in the absence of extensive resources, often resorting to innovative techniques, data augmentation, and domain adaptation. The resource-constrained text classification is divided into two subcategories, i.e., non-Bengali and Bengali, which will be described in the following.

2.6.2.1 Non-Bengali Text Classification

Limited research has been conducted on resource-constrained languages other than Bengali, such as Hindi, Arabic, Tamil, Italian, and others. Joshi et al. [73] developed a CNN-based model with FastText embeddings for Hindi text classification, achieving an accuracy of 92.8% across six document categories. However, it's worth noting that many previous approaches in low-resource languages, including Bengali, have encountered challenges related to out-of-vocabulary issues

and the lack of consideration for sub-word information, which is crucial for improving classification performance.

Patwa et al. [5] created a dataset of hostile posts in Hindi and developed a system to identify such posts in the online space. They utilized m-BERT embeddings and employed the LibSVM classification method to distinguish between hostile and non-hostile posts. Their system achieved a maximum accuracy of 84.11% for coarse-grained classification. Hussein et al. [84] designed a system for detecting Covid infodemic content in Arabic using tweets. This system can classify seven predefined queries based on a dataset consisting of 2,556 Arabic tweets. They achieved a maximum accuracy of 67.7% by utilizing the AraBERT framework. Mattern et al. [85] curated a corpus of German Covid-related fake news, encompassing 28,056 actual news pieces and 13,186 fake news articles. Their BERT + Social context system attained a peak accuracy of 82.40% when applied to this dataset. In the context of Persian fake news detection, a classification method based on LibSVM was explored, resulting in a maximum accuracy of 87.00% [86]. Harakawa et al. [87] developed a keyword extraction method for Japanese text on Twitter. However, this method primarily focused on word-level features and did not consider sentence-level linguistic semantics.

In [83], an authorship attribution (AA) system was developed for Latin text using Bray–Curtis distance and logistic regression, achieving a remarkable maximum accuracy of 99.86%. However, this system does not consider higher-dimensional data or incorporate deep or transformer-based learning techniques. Anwar et al. [88] introduced an unsupervised machine learning-based AA system for Urdu text, employing methods like Latent Dirichlet Allocation (LDA). This system achieved an F_1 score of 92% when applied to a dataset comprising 6,000 texts. Jambi et al. [89] designed an AA system for Arabic short-microblog texts using SVM and KNN approaches. The classifier achieved a maximum accuracy of 65.00% when tasked with attributing authorship to 1,000 authors based on 1.5 million Arabic tweets. In another study [90], statistical classifiers such as Bagging and AdaBoost, coupled with ensemble methods, were employed for Arabic authorship attribution. This method utilized stylometric features and was evaluated on a self-built dataset. Taha et al. [91] explored a two-level AA learning system for Arabic author texts. The system initially extracted vocabulary-level features, which were then fed into the second learning phase. However, it's worth noting that vocabulary-level statistical features may not capture word semantics effectively. Furthermore, Anwar et al. [88] developed a method based on word n-grams and LDA for Urdu authorship attribution, achieving an impressive F_1 score of 92.00% on a dataset comprising 6,000 Urdu newspaper articles.

Research on resource-constrained languages other than Bengali, such as Hindi, Arabic, Tamil, Italian, and others, remains limited. While some efforts have been made, challenges like out-of-vocabulary problems and sub-word information consideration hindered classification performance in low-resource languages. Overall, further research is needed to address these languages' unique challenges and boost classification performance.

2.6.2.2 Bengali Text Classification

Bengali characterized as a low-resource language with limited linguistic resources and inadequate corpora, has seen relatively nascent developments in automatic text document classification [92, 48]. Earlier approaches included [4], who employed the TF-IDF technique to extract features for Bengali document categorization. Their SVM classifier was trained on 900 text documents and tested on 100 documents across five categories. Similarly, [93] introduced a TF-IDF feature extraction method coupled with a gradient descent classifier for Bengali document classification, achieving an accuracy of 93.78%. An alternative approach involved using stochastic gradient descent (SGD)-based classifiers for Bengali text categorization [94], achieving a 93.33% test accuracy across nine distinct categories with 10,000 training and 4,651 testing document samples. However, these SGD-based techniques exhibited limitations when dealing with a limited number of text categories and test samples, particularly when confronted with high-dimensional feature spaces [95].

Recent advancements introduced Word2Vec-based feature extraction techniques, combining K-Means clustering with SVM for Bengali text classification [96]. Leveraging 19,705 online news text documents spanning seven text categories, this approach excelled at extracting semantic features from a large vocabulary. However, it faced challenges in handling outlier data due to the K-Means algorithm. Another innovative solution came from [97], who developed a Bengali document categorization system based on Word2Vec and Deep CNNs. With a training dataset of 86,199 news text documents spanning 12 document categories, the system achieved an impressive average document classification accuracy of 94.96% across 10,707 test samples. Nevertheless, this system did not effectively capture sentence-level semantic and syntactic meanings. Furthermore, [98] proposed a graph-based feature extraction technique paired with an LSTM RNN-based classifier for Bengali text categorization, focusing on nine document classes and attaining a remarkable 99.21% classification accuracy across 14,371 text documents. However, feature extraction in this system proved time-consuming and fell short in capturing semantic and syntactic features effectively.

Bengali Authorship Attribution (AA) research have shifted towards Deep Learning (DL) methodologies to address limitations associated with traditional Machine Learning (ML) approaches. Khatun et al. [99] delved into AA using character-level embeddings and Convolutional Neural Networks (CNN). Their method achieved peak accuracies of 98.00% and 81.2% for the BAAD6 and BAAD14 datasets, respectively. However, character-level embeddings struggled to capture word-level semantic and syntactic information, and the CNN hyperparameters remained untuned. Fusion-based approaches combining CNN and Artificial Neural Networks (ANN) have been introduced in [100]. They reported a maximum accuracy of 80.74% for the Bengali-Quora corpus (BQC) using ANN, 96.71% for BAAD6 using CNN+ANN, and an impressive 99.78% for BAAD16 using the ANN method. It's important to note that their classification frameworks (both ANN and CNN) lacked hyperparameter tuning tailored to Bengali corpora. Additionally, [101] employed a frequency-based GloVe approach in conjunction with tuned CNN (GloVe+CNN). This method achieved peak accuracies of 93.45%, 95.02%, and 98.67% for the BACC-18, BAAD16, and LD datasets, respectively. However, it's worth noting that manual hyperparameter tuning of the CNN model required significant time and memory resources.

A transformer-based approach for Bengali text classification was introduced by [102], showcasing remarkable results with an accuracy of 96.39% on the ProthomAlo corpus, 93.05% on the BARD corpus, and 78.53% on the IndicNLP corpus. It's important to note that their model did not undergo hyperparameter tuning specific to Bengali corpora. However, it's worth highlighting that the majority of text classification models, whether in low or high-resource languages, have generally not been fine-tuned for domain-specific corpora, and they often grapple with the challenge of handling Out-of-Vocabulary words. Meanwhile, [103] developed a text classification system for the low-resource Assamese language, achieving a commendable accuracy of 89.3% on a dataset they constructed themselves. Nevertheless, fusion-based text classification systems have primarily focused on high-resource languages, creating a research gap for low-resource languages like Bengali.

In the context of Bengali text document classification, there have been advances and limitations. Traditional approaches like TF-IDF and stochastic gradient descent (SGD) showed promise but had challenges dealing with diverse categories and high-dimensional features. Recent methods, including Word2Vec-based techniques and deep learning models, improved semantic feature extraction but struggled with outlier data and sentence-level semantics. In Bengali Authorship Attribution (AA), Deep Learning (DL) methods like character-level

embeddings and fusion models demonstrated potential but often lacked customized hyperparameter tuning. A GloVe-based approach achieved impressive results but required manual tuning. A transformer-based approach yielded excellent accuracy without Bengali-specific tuning. Overall, most text classification models, whether for low or high-resource languages, face challenges like Out-of-Vocabulary words and lack fine-tuning for specific domains. While low-resource languages like Assamese received some attention, fusion-based text classification remains predominantly focused on high-resource languages, creating opportunities for further research.

2.7 Research Gaps

In investigating prior research on text classification in both high-resource and resource-constrained languages, this study identifies notable research gaps. Specifically, Bengali, with its intricate morphology and linguistic idiosyncrasies that set it apart from high-resource languages, presents distinct challenges in the realm of text classification. Unlike high-resource languages, Bengali cannot readily leverage existing resources such as corpora, pre-trained embeddings, or well-tuned classification models. Previous studies in Bengali text classification predominantly relied on traditional techniques like SVM, SGD, KNN, and Deep CNNs, typically addressing a limited range of document categories and working with smaller corpora. Additionally, existing approaches to Bengali Authorship Attribution (AA) predominantly utilized non-contextual methods such as TF-IDF, N-Gram, GloVe, FastText, Word2Vec, and Character-embedding, which failed to effectively handle Out-of-Vocabulary (OOV) and context-awareness issues, limiting their ability to capture comprehensive semantic and syntactic meanings. Previous research primarily focused on non-contextual features in the realm of attentional fusion-based Bengali text classification, resulting in challenges related to OOV words and suboptimal hyperparameter tuning, particularly for domain-specific news classification corpora. While transformer-based language models were explored in some studies, they often lacked early fusion techniques, leading to difficulties in extracting context and knowledge-aware word-level features and neglecting OOV word concerns. Furthermore, hyperparameter tuning strategies designed for high-resource languages did not yield optimal results in the case of low-resource languages like Bengali, where morphological variations presented challenges for both monolingual and multilingual transformer models.

To address these research gaps comprehensively, this research introduces a novel Bengali text classification system that utilizes an attention-based feature

fusion network of transformers. **This system innovatively addresses these research gaps in five key ways**:

(i) Active Learning for Resource-constrained Corpora Development: This study pioneers an active learning-based approach for constructing corpora in resource-constrained environments, effectively alleviating data scarcity challenges for low to zero-resource languages.

(ii) Advanced OOV Word Handling: By capitalizing on Byte-Pair Encoding and WordPiece tokenization within transformer-based models, this research enables the extraction of Out-of-Vocabulary (OOV) words while capturing context and knowledge-rich features.

(iii) Tuned Hyperparameters for Resource-constrained Contexts: A meticulously designed hyperparameter tuning strategy, combining grid search and Ray Tune, is introduced to cater specifically to the needs of low-resource languages.

(iv) Innovative Attention-Based Early Fusion: Novel attention-based early fusion models seamlessly integrate monolingual and multilingual classification features, enhancing the text classification process.

(v) Effective Mitigation of Class-Imbalance Impact: Through the implementation of focal loss techniques tailored to class-imbalanced corpora, this research significantly enhances performance, particularly in classes with limited sample sizes.

These distinctive contributions center on Bengali text classification, harnessing a dedicated language model and cutting-edge methodologies to elevate performance while mitigating the limitations that have hindered prior research endeavors.

2.7.1 Research Questions and Answers

The primary objective of this research is to reduce the gap between high-resource and resource-constrained languages, enabling the utilization of high-resource tools and adapting their hyperparameters to develop real-time text classification applications for low-resource scenarios. In essence, this research aims to address a series of **Research Questions (RQs)** sought to gain a comprehensive understanding of the effectiveness and functionality of resource-constrained text classification systems.

- **RQ1:** How can active learning strategies enhance corpus collection efficiency for resource-constrained languages?

- **RQ2:** How can the challenges posed by Out-of-Vocabulary (OOV) words be mitigated, and what methods can enhance the extraction of context and knowledge-aware features to elevate Bengali text classification performance?
- **RQ3:** What approaches are optimal for fine-tuning hyperparameters in transformer-based language models to cater specifically to low-resource settings and diverse corpora?
- **RQ4:** What is the most effective way to integrate distinct features derived from multiple models to enhance Bengali text classification performance?
- **RQ5:** How can the performance of Bengali text classification be enhanced, particularly concerning class-imbalance issues within the corpus?"

The significant contributions and corresponding Answers to the Research Questions (ARQs) are succinctly summarized as follows:

- **ARQ1:** Introduction of an active learning-based semi-automated corpus development (ALSCD) framework for constructing corpora in resource-constrained languages, effectively applied to acquiring labeled and unlabelled Bengali corpora. The ALSCD framework notably reduces annotation time and corpus construction costs. (i.e., Section 3.3)
- **ARQ2:** Utilization of transformer-based models alongside Byte-Pair Encoding (BPE) and WordPiece tokenization to generate contextual word embeddings for Out-of-Vocabulary (OOV) words. This approach harnesses transformer models' capabilities to enhance context and knowledge-aware features, significantly improving Bengali text classification performance (as demonstrated in Section 5.4.3.6 and Table 6.22, Table 6.30, Table 6.46, Table 6.38).
- **ARQ3:** Implementation of a hyperparameter tuning strategy combining Grid Search (GS) and Ray Tune (RT) algorithms. This approach identifies influential hyperparameters specific to the given corpora, determines optimal value ranges, and validates selected hyperparameter values across multiple corpora (Details in Section 5.3.1, Algorithm 9 and Section 5.4.1, Algorithm 10 and Table 6.19).
- **ARQ4:** Development of an attention-based early fusion technique that selects the top-performing four models based on late fusion accuracy. This

method integrates the attention-based early fusion approach into these chosen models, effectively fusing features from multiple models to enhance text classification performance (i.e., Section 5.4.3).

- **ARQ5:** Implementation of the focal loss on class-imbalanced corpora, prioritizing to minimize the loss for categories with fewer samples while reducing the impact of loss for categories with larger sample sizes. The use of focal loss enhances classification performance for class-imbalanced corpora (Details in Section 6.5.5, Table 6.33).

2.8 Problem Formulation

The aim of this study is to design a text classification system with a specific focus on resolving the problems associated with resource-constrained text classification. The primary problems encountered in resource-constrained text classification involve inadequate corpora, pre-trained embeddings, and well-tuned pre-trained classification models. To overcome these problems, we have devised a resource-constrained text classification system as illustrated in Figure 2.1. This system is structured around three key phases: (i) corpora development, (ii) embedding model development, and (iii) classification model development. The following describes each of the phases.

(i) Corpora Development The phase of corpora development plays a pivotal role in the resource-constrained text classification system. In this study, we employ two distinct approaches, namely a manual annotation-based approach and an active learning-based approach, to gather embedding and classification corpora. Both of these approaches initiate by selecting the source text and generating a list of relevant URLs. This list of URLs serves as the input for the corpora development stage. Within this stage, a specialized web crawler is designed to adhere to the "robots.txt" policy and retrieve all pertinent texts. These crawled texts then undergo preprocessing through a dedicated text preprocessing module. Subsequently, the preprocessed texts are divided, with one set serving as the basis for embedding corpora, while the remaining texts are directed to the manual annotation or active learning-based approach for annotation verification and quality assessment. As part of this research, we have created five embedding corpora, denoted as $\{BEC, WEC, CovEC, ECoVE, AraEC\}$. Simultaneously, the manual annotation and active learning-based approaches have been employed to establish seven classification corpora, labeled as $\{BACC -$

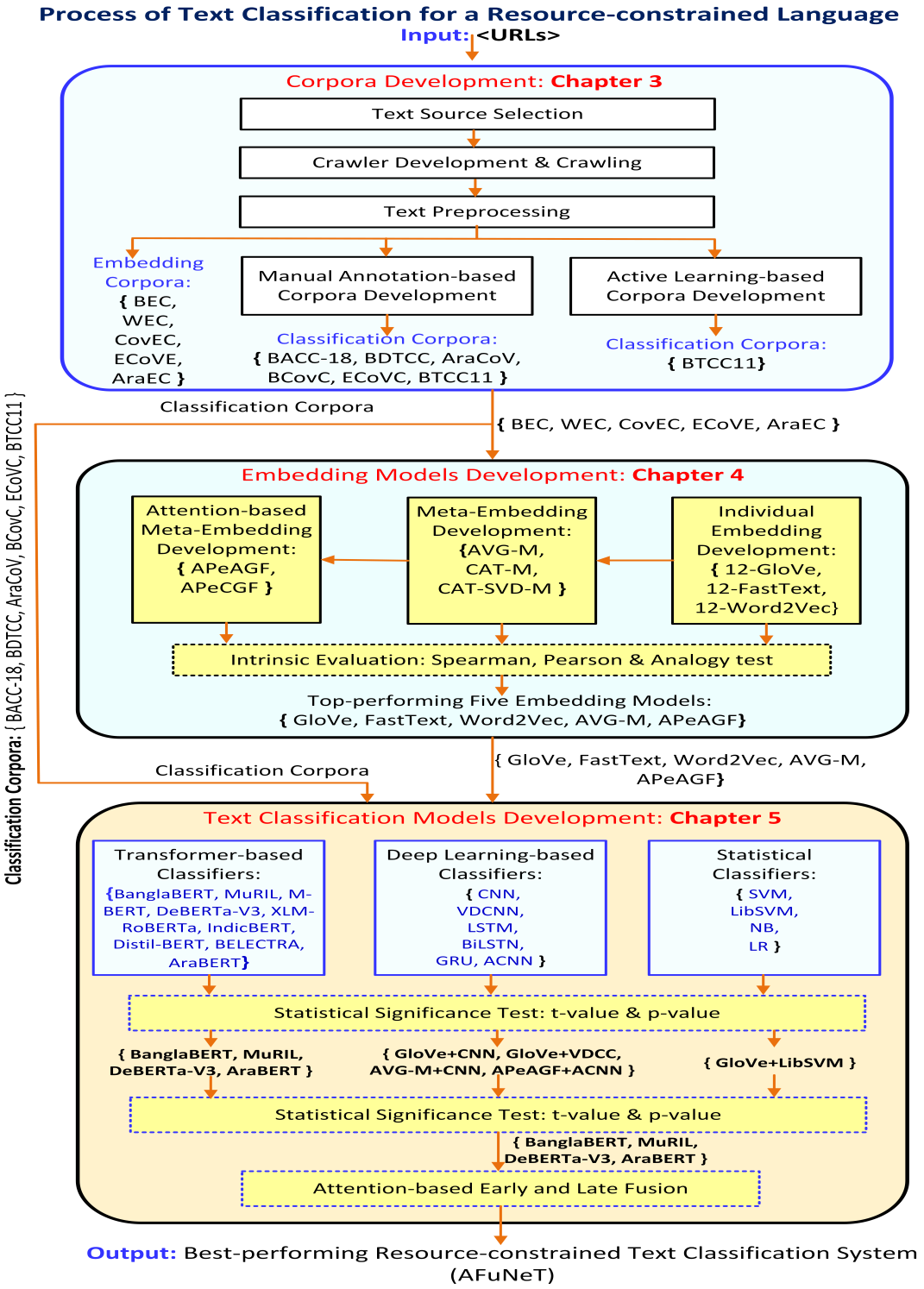


Figure 2.1: Abstract view of text classification for a resource-constrained language

18, $BDTCC$, $AraCoV$, $BCovC$, $ECoVC$, $BMMTC$, $BTCC11\}$. The comprehensive details of the corpora development process will be presented in **Chapter 3**.

(ii) Embedding Models Development In this phase, the module for developing embedding models takes a set of embedding corpora, which includes BEC, WEC, CovEC, ECoVE, and AraEC, as inputs. The module's outputs comprise the best-performing embedding models, specifically three top-performing individual embeddings, three meta-embeddings, and three attention-based meta-embeddings. To begin, three different embedding model generation methods, namely GloVe, FastText, and Word2Vec, are employed to create individual embeddings. These individual embeddings undergo hyperparameter tuning using the "EPI" Algorithm presented in Algorithm 5. Furthermore, meta-embeddings and attention-based meta-embeddings are derived from individual embeddings. Meta-embeddings combine the features from multiple individual embedding models to create generalized feature vectors. The quality of these embeddings, encompassing individual embeddings, meta-embeddings, and attention-based meta-embeddings, is assessed through intrinsic evaluations. The top three performing individual embeddings (GloVe, FastText, Word2Vec), one meta embedding (AVG-M), and one attention-based meta-embedding (APeAGF) are selected for each of the embedding corpora. These best-performing embedding models are subsequently forwarded to the text classification models development phase. Further details regarding the development of embedding models can be found in **Chapter** 4.

(iii) Classification Models Development The goal of this research is to develop a text classification system focusing on the Bengali language and evaluate its performance using a set of corpora denoted as C_i, where $i \in \{BACC-18, BDTCC, AraCoV, BCovC, ECoVC, BTCC11, OSBC, BARD, IndicNLP, ProthomAlo\}$. Each corpus, C_i, is randomly split into training (70%) and inference sets (30%), represented by C_i^{tr} and C_i^{ts}, respectively. In this study, we have used four open-access Bengali corpora, such as BARD [104], OSBC[1], IndicNLP [105], and ProthomAlo[2]. The text space for training and testing is denoted as X_i^{tr} and X_i^{ts}, respectively, while the corresponding label space is represented by Y_i^{tr} and Y_i^{ts} for corpus i.

In this research, three major text classification techniques were employed: statistical classifiers (SVM, NB, RF), deep learning-based classifiers (CNN, VDCNN, ACNN, LSTM, BiLSTM, GRU), and transformer-based classifiers (BanglaBERT, MuRIL, DeBERTa-V3, XML-RoBerTa, IndicBERT, DistilBERT, M-BERT, AraBERT). All classification models had their hyperparameters tuned using the

[1]https://www.kaggle.com/datasets/ratulcse10/open-source-autonomous-bengali-corpus
[2]https://www.kaggle.com/datasets/twintyone/prothomalo

text classification corpora and were generated with the tuned hyperparameters. A statistical significance test, which includes t-values and p-values from Welch's t-test and Friedman's test, was applied to the classifiers, resulting in the selection of four top-performing transformer-based classifiers: BanglaBERT, MuRIL, DeBerTa-V3, and AraBERT. For deep learning, GloVe+CNN, GloVe+VDCNN, AVG-M+CNN, and APeAGF+ACNN classifiers were chosen, and GloVe+LibSVM was selected for statistical classifiers. Subsequently, an inter-classifier statistical significance test was conducted, leading to the selection of the top four classifiers: BanglaBERT, MuRIL, DeBERTa-V3, and AraBERT. Using both late and early fusion techniques, we combined BanglaBERT and MuRIL for Bengali text classification corpora and DeBERTa-V3 and AraBERT for Arabic text classification tasks. The resulting attention-based early fusion model, combining BanglaBERT and MuRIL, is known as AFuNeT, our proposed model.

The development of AFuNet involves four crucial steps in this study. In the **first step**, the baseline transformer-based language models (TL) are evaluated with the default hyperparameters H_j^I using Eq. 2.1.

$$\begin{aligned} TL_{ji}(H_j^T) = \Psi_L(TL_j, H_j^I, C_i), i \in \{&BTCC11, OSBC, BARD, IndicNLP, \\ &ProthomAlo,\}, j \in \{BanglaBERT, BELECTRA, MuRIL, \\ &DeBERTa, MBERT, XLMRoBerta, IndicBERT, DistilBERT\} \end{aligned} \tag{2.1}$$

Here, $TL_{ji}(H_j^T)$ denotes the fine-tuned model with tuned hyperparameters H_j^T of model TL_j and corpus C_i. The $\Psi_L(.,.,)$ indicates the function for TL-based baseline models evaluation and hyperparameters tuning. In the similar, Deep learning-based baseline models are evaluated and tuned for their hyperparameters using Eq. 2.2.

$$\begin{aligned} DL_{kli}(H_k^T) = \Psi_D(DL_k, H_k^I, nED_l, C_i), k \in \{&CNN, LSTM, BiLSTM, GRU\}, \\ &l \in \{GloVe, FastText, Word2Vec\} \end{aligned} \tag{2.2}$$

Here, $DL_{kli}(H_k^T)$ indicates the deep learning-based tuned model with tuned hyperparameters H_k^T of model k, non-contextual embedding l, and corpus i. The $\Psi_D(.,.,.)$ denotes the evaluation and hyperparameter tuning function for DL-based baseline models. H_k^I denotes the default hyperparameters of DL-based model k. The nED_l denotes the non-contextual embedding models l. Now, the best-performing 8 models are selected from $TL_{ji}(H_j^T)$ and $DL_{kli}(H_k^T)$ using Eq.

2.3.

$$B_m = \Psi_V(TL_{ji}(H_j^T), DL_{kli}(H_k^T), Loss_z), m = 1, ..., 8, z = \{L^{Fo}, L^{Cs}\} \quad (2.3)$$

Here, B_m denotes the best-performing model's index by m, and $\Psi_V(.,)$ represents the model evaluation and selection function. The variable $Loss_z$ denotes the loss calculation method, where z can take on one of two methods: L^{Fo} for focal loss and L^{Cs} for categorical cross-entropy loss.

In **the second step**, the tuned models generate late fusion-based statistics (i.e., accuracy, precision, recall, and F1-score) by the Eq. 2.4.

$$FM_{pz} = \Phi_{pz}(B_m, LF_d), pz = 1, ..., \binom{8}{3} \times |d|, d \in \{max - voting, averaging\} \quad (2.4)$$

In this context, FM_{pz} represents the late fusion-based statistics indexed by pz, and $\Phi_{pz}(.,.)$ denotes the late fusion function. The variable LF_d refers to the late fusion method d, which can be either *max-voting* or *averaging*. The term $\binom{8}{3}$ corresponds to the total number of combinations, and $|d|$ represents the set size of late fusion methods. This step combines the predictions from B_m and LF_d to create various late fusion statistics, each aiming to enhance classification performance through late fusion methods. Now, Eq. 2.5 selects the best-performing four models based on accuracy.

$$RM_r = \underset{z1=1}{\overset{(pz+|m|)\times|i|}{\arg\max}}[\lambda_{WF}(FM_{pz}, B_m)], r = 1, ..., 4 \quad (2.5)$$

where RM_r represents the best-performing model index by r in the list, and the function $\underset{z1=1}{\overset{(pz+|m|)\times|i|}{\arg\max}}[.]$ is responsible for ranking the models based on accuracy value. The $|m|$ denotes the total number of best-performing models. In contrast, $|i|$ represents the corpora set size , The function $\lambda_{WF}(FM_{pz}, B_m)$ is used to selects best-performing 4 models. Index 1 corresponds to the best-performing model, with subsequent indices indicating subsequent rankings in order of models.

In the **third step**, four best-performing models produce attention-based early fusion models using Eq. 2.6 and select the best model.

$$\Delta AFuNet = \underset{z2=1}{\overset{\binom{|r|}{4}\times\binom{|r|}{3}\times\binom{|r|}{2}\times|i|\times|n|\times|q|}{\arg\max}}[\lambda_{WF}(\eta_q(RM_r, C_i), ST_n)], \quad (2.6)$$
$$q \in \{cat, avg, acat, aavg\}, n \in \{Welch'st - test, Friedman - test\}.$$

$\Delta AFuNet$ is the best-performing attention-based early fusion model. The func-

tion $\arg\max_{z2=1}^{\binom{|r|}{4}\times\binom{|r|}{3}\times\binom{|r|}{2}\times|i|\times|n|\times|q|}[.,]$ responsible for selecting the best model based on statistical significance test. The $|r|$ denotes the total number of best-performing models, $|q|$ denotes the number of early fusion-based methods, and $|n|$ represents the total number of statistical tests. The function $\eta_q(.,.,)$ generates attention-based early fusion models using the combination of $\binom{|r|}{4} \times \binom{|r|}{3} \times \binom{|r|}{2} \times |i| \times |n| \times |q|$ and measures statistical significance tests with method n.

In the **fourth step**, ablation studies were conducted on $\Delta AFuNet$ based on dropout value (Dr), number of attention layers (NAL), and attentional embedding dimension (AED) using Eq. 2.7.

$$AFuNet = \delta(\Delta AFuNet, C_i, \alpha_c), c \in \{Dr, NAL, AED\} \tag{2.7}$$

Equation 2.7 defines $AFuNet$ as the resulting attention-based early fusion network, with $\delta(.,.,)$ representing the function governing ablation studies. We analyze the effects of Out-of-vocabulary (OOV) words, class-imbalance issues, and the choice between monolingual and multilingual language models on the performance of $AFuNet$. Through this analysis, we ascertain the model's robustness and adaptability. Ultimately, $AFuNet$ emerges as the refined and best-performing development model for Bengali text classification, showcasing its exceptional capabilities across various corpora ($C_i|i \in \{BTCC11, OSBC, BARD, IndicNLP, ProthomAlo\}$).

2.9 Summary

Chapter 2 explores essential text classification concepts and an overview of text classification technologies. Within the related works section, a comprehensive examination of prior research in both high-resource and resource-constrained text classification is conducted. This review delves into their findings, highlights, limitations, and key contributions. The chapter then pivots towards identifying research gaps and proposing solutions. These solutions form the basis for research questions and their potential answers. Finally, the problem formulation section outlines the detailed process of developing a resource-constrained text classification system. Subsequent chapters will delve into the specifics of each major contribution, encompassing corpora development, embedding model development, and classification model development.

Chapter 3

Corpora Development

Corpora are extensive and structured collections of textual datasets, encompassing various types of content like Social media posts, Online newspapers, blogs, and textual conversational platforms. They play a fundamental role in linguistic analysis, language research, and the development of language-related applications. In the context of resource-constrained scenarios, creating corpora is a pivotal and initial step for subsequent tasks. This chapter introduces corpora development techniques and presents a system for constructing corpora tailored to resource-constrained text classification.

3.1 Corpora Development Techniques

Resource-constrained corpora development encompasses two primary techniques: manual annotation-based and active learning-based approaches. Manual annotation-based techniques involve human experts selecting specific text sources and generating lists of URLs for data collection. These lists serve as input for the corpora development step, where custom web crawlers are designed to retrieve text content, adhering to the "robots.txt" policy. The collected texts are subsequently subjected to text preprocessing. To ensure the quality and accuracy of annotations, a subset of the preprocessed texts is routed to the manual annotation process, which is guided by expert input. Active learning-based techniques use a similar initial URL list but leverage machine learning to iteratively select texts for manual annotation. This iterative process enables the system to focus on uncertain or challenging examples, making the best use of human annotators' time and expertise. These resource-constrained corpora development techniques are crucial for creating high-quality datasets for text classification tasks in languages with limited resources. An abstract view of corpora development is depicted in Fig. 3.1. The following subsections provide detailed explanations of each of the

corpus development methods.

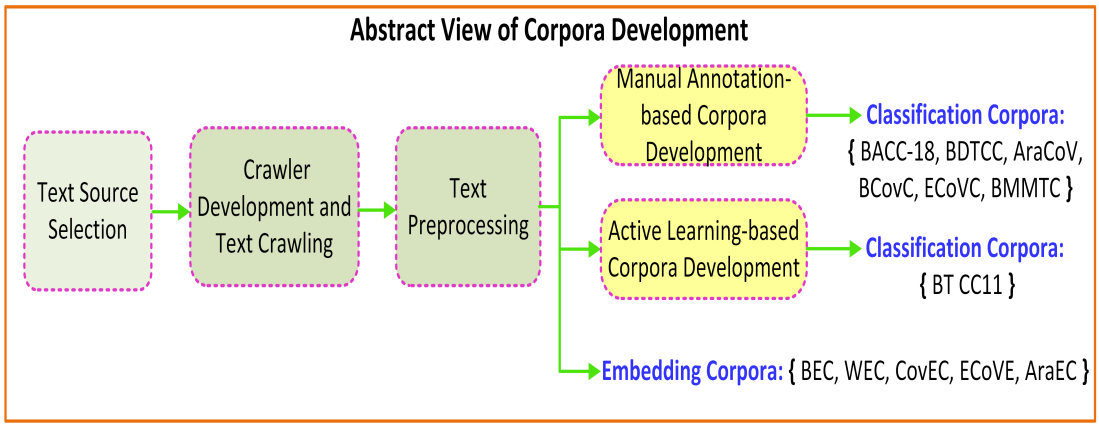


Figure 3.1: Abstract view of corpora development

3.1.1 Texts Source Selection

To adhere to copyright policies, the process of crawling low-resource text content begins with an examination of the robot.txt policy of potential data sources to determine permissible web URLs. This initial step takes a set of input URLs and outputs a refined set of URLs eligible for data crawling. In this study, data was sourced from diverse platforms, which is summarized in Table 3.1. The embedding corpora are sourced from the identical URLs corresponding to their respective languages. As indicated in Table 3.1, the BACC-18 corpus is gathered from the repositories of Blogs, NLTR, and the Bangla Academy. In contrast, the BDTCC, BTCC11, BCovC, and BMMTC corpora are constructed using content from online newspapers, blogs, and online TV news portals. The English and Arabic COVID-19 text identification corpora are derived from online newspaper archives. All of this data spans from October 9, 2018, to December 10, 2022.

3.1.2 Crawler Development and Text Crawling

In this phase, the URLs list produces a list of crawled text. The Arabic crawler is developed based on Scrapy library [110] and defined in the following. The Scrapy-based link extractor extracts the corresponding link content based on pre-defined rules date robot.txt policy.

1. Import required libraries:
    - requests
    - BeautifulSoup

Table 3.1: Summary of corpora source text

Corpus	Text Source (URLs/Ref.)	Language	Domain
BACC-18	https://www.nltr.org/	Bengali	NLTR
	http://banglaacademy.gov.bd/	Bengali	Bangla Academy
	Git repositories [106]	Bengali	Blogs
	Online Blogs [107, 108, 109]	Bengali	Blogs
BDTCC, BTCC11, BCovC, BMMTC	https://www.prothomalo.com/	Bengali	Newspaper
	https://www.anandabazar.com	Bengali	Newspaper
	http://www.bigganprojukti.com	Bengali	Blogs
	https://cadetcollegeblog.com	Bengali	Blogs
	https://blog.mukto-mona.com	Bengali	Blogs
	https://dainikazadi.net/	Bengali	Newspaper
	http://banglanews24.com/	Bengali	Newspaper
	http://www.banglatribune.com/	Bengali	Newspaper
	https://www.dailyinqilab.com/	Bengalu	Newspaper
	https://kalerkantho.com/	Bengali	Newspaper
	https://www.ekushey-tv.com/	Bengali	Online-TV News
	http://www.banglatribune.com/	Bengali	Newspaper
AraCoV	https://www.akhbarona.com	Arabic	Newspaper
	https://www.alarabiya.net	Arabic	Newspaper
	https://www.albayan.ae	Arabic	Newspaper
	https://www.alittihad.ae	Arabic	Newspaper
	https://www.alkhaleej.ae	Arabic	Newspaper
ECoVC	https://abcnews.go.com/	English	Newspaper
	https://www.aljazeera.com/	English	Newspaper
	https://www.thedailystar.net/	English	Newspaper
	https://www.foxnews.com/	English	Newspaper
	https://gulfnews.com	English	Newspaper
	https://www.arabnews.com	English	Newspaper

2. Define a function to crawl a web page:
   (a) Send an HTTP GET request to the specified URL.
   (b) Check if the request was successful (status code 200).
   (c) If successful:
      - Parse the page content using BeautifulSoup.
      - Extract and process data from the page.
      - Save or process the extracted data as needed.
   (d) If unsuccessful or an exception occurs, handle the error.
3. In the main function:

(a) Specify the URL to start crawling from.

(b) Call the crawl_page function with the start URL.

(c) Check if the crawl was successful and print the result.

This algorithm outlines the essential steps for web data crawling using Python. It starts by importing necessary libraries like requests and BeautifulSoup. The main function defines the process to crawl a web page. It sends an HTTP GET request to a specified URL, checks for a successful response (status code 200), and, if successful, uses BeautifulSoup to parse and extract data from the page. In the main part of the algorithm, you specify the URL to begin crawling. The crawl function is called with this starting URL. If the crawl is successful, the algorithm processes or saves the extracted data as required. In case of errors or exceptions, error handling is incorporated. This algorithm provides a foundation for extracting information from web pages systematically.

3.1.3 Text Preprocessing

The input is a set of noisy and unlabelled texts, and the output is a set of preprocessed texts. Sometimes, the crawled text is noisy and ill-formatted due to the web's freestyle writing. Thus, initially apply the following four steps to remove the noisy or ill-formatted characters from the texts:

- Step1: Remove foreign alphabets and digits with a NULL value.
- Step 2: A single white space is replaced with regular expressions and symbols.
- Step 3: HTML tags, hashtags, URLs, and punctuation are replaced by white space.
- Step 4: Replaced a single new line with multiple new lines.

As a result of these steps, some text files may become empty. To maintain data quality, any empty text files and those containing fewer than five Arabic words are removed from the corpus. Additionally, any text files exceeding 5 KB in size are also excluded. This process ensures that the final corpus is free from noisy or low-quality text data. *Following the preprocessing steps, half of the texts are randomly selected for the purpose of building an embedding corpus, while the remaining texts are allocated for the development of a text classification corpus. Those texts designated for text classification corpus development undergo further stages involving annotation, verification, and quality assessment.*

3.2 Manual Annotation-based Corpora Development

The manual annotation-based corpora development approach involves the creation of labeled datasets through human annotation. This process necessitates domain experts or annotators to assess and categorize the text data, assigning appropriate labels or tags. As a result of this manual annotation process, this research produced six distinct corpora: BACC-18 (Bengali Authorship Classification Corpus), BDTCC (Bengali Text Classification Corpus), AraCoV (COVID-19 Text Classification Corpus for Arabic), BCovC (Bengali COVID Text Classification Corpus), ECoVC (English COVID-19 Text Classification Corpus), and the Bengali Multi-Modal Text Classification Corpus. The development of these corpora entails the following steps:

3.2.1 Annotation, Verification and Quality Measures for Text Data

In the data annotation phase, unlabelled texts are provided as input, and the objective is to assign predefined labels to them with the assistance of human annotators. This annotation process involves two human annotators who follow an empirical rule. According to this rule, if both annotators reach a consensus on the category of a text, it is then assigned a tag from the predefined categories. However, in cases where there is a discrepancy between the annotators regarding the appropriate category for a text, it is flagged for the verification phase. During the annotation process, annotators quickly review articles and attribute a category based on the semantics of the text. To aid them in this task, an annotation tool built in C# for Windows is utilized.

In the expert-level verification phase, the texts undergo a rigorous examination by a linguistic expert. The linguistics expert thoroughly reads each text and assesses whether it exhibits semantic or syntactic similarity to any of the predefined classes. If a text aligns with one of these classes, it is deemed suitable for inclusion in the final corpus. On the other hand, if a text does not exhibit such alignment, it is excluded from the corpus.

3.2.2 Annotation, Verification and Quality Measures for Image-Text Data

The dataset for Bengali multimodal text classification underwent meticulous preparation to ensure its quality and reliability. The process involved collecting image-text multimodal data relevant to the classification task. Subsequently, each sample was meticulously annotated by two annotators. In the event of any disagreement between annotators, a verified expert intervened to resolve discrepancies and ensure accuracy.

Furthermore, special attention was given to maintaining consistency and coherence across the dataset. This included standardizing annotation guidelines and procedures to minimize variability in labeling. Additionally, rigorous quality control measures were implemented to identify and rectify any errors or inconsistencies in the dataset. The dataset preparation phase was conducted with meticulous care and attention to detail, aiming to provide a high-quality resource for training and evaluating Bengali text classification models. The thorough annotation process, supplemented by expert verification, ensures the reliability and integrity of the dataset, laying a solid foundation for subsequent research and experimentation in the field.

During the verification step, ten experts conducted a thorough review of the crowd-sourced corpus. The level of agreement among annotators was assessed using Cohen's kappa, a measure of how well one annotator's judgments align with those of another annotator [111]. To evaluate the standard inter-annotator agreement, we calculated a pairwise kappa coefficient using Equation 3.1:

$$K = \frac{P_o - P_e}{1 - P_e} \tag{3.1}$$

Here, P_o represents the relative observed agreement, and P_e indicates the hypothetical probability of chance agreement.

3.2.3 Corpora Statistics

Two types of corpora have developed using the manual annotation approaches such as (i) Embedding corpora and (ii) text classification corpora. The subsequent paragraphs provide insights into the statistics of the embedding corpora and text classification corpora.

(i) Embedding Corpora Statistics In this study, we created five embedding corpora, specifically for Bengali, English, and Arabic texts.

Bengali Embedding Corpus (BEC): Table 3.15 presents statistics for the Bengali Embedding Corpus (*BEC*).

Table 3.2: Bengali embedding corpus (*BEC*) statistics

Attributes	Values
No. of unlabelled texts	1,129,407 (1.14 M)
No. of sentences	364,055,081 (364.05 M)
No. of words	1,146,833,902 (1146.83 M)
Maximum word frequency	11,737,198 (11.73 M)
No of unique words	10,867,699 (10.86 M)
No. of unique words with min. frequency 2	2,867,701 (2.86 M)
Maximum no. of sentences	376
Minimum no. of sentences	1
Maximum text size	107.884 KB
Minimum text size	0.048 KB

To our knowledge, this is the most extensive Bengali unlabelled corpus, characterized by its substantial diversity in unique words, boasting an impressive 10.86 million unique words. The corpus comprises 1.14 million (M) unlabelled texts, encompassing 364.05 million different sentences. The document size within the corpus varies, with the largest document measuring 107.884 KB and the smallest being 0.048 KB.

Word Embedding Corpus (WEC): The Word Embedding Corpus (WEC) is a compilation of unlabelled texts acquired by crawling various Bengali newspaper portals, serving as a primary input for the embedding algorithm. This corpus is instrumental in generating lower-dimensional feature vectors for input words, which play a crucial role in authorship classification. Data was crawled between January 1, 2015, and December 30, 2020. We ensured compliance with newspaper crawling policies, which permit automatic crawling via robots.txt. Legal restrictions on crawling were also verified. Table 3.3 presents key statistics for WEC, which consists of 0.95 million unlabelled texts, encompassing a total of 854.16 million words, including 2.75 million unique words.

Table 3.3: Key Statistics of Word Embedding Corpus (WEC)

Attributes	Values
Total no. of texts (M)	0.95
Total no. of sentences (M)	28.47
Total no. of words (M)	854.16
Total no. of unique words (M)	2.75
Total no. of unique words at min count 2 (M)	1.97

Covid Embedding Corpus(CovEC): The *CovEC* is an unlabeled corpus designed for the purposes of generating single embedding models, conducting evaluations, and identifying Covid-related text. It is an extension of the previously established Bengali embedding corpus (*EC*) [2]. Table 3.4 shows the key statistics of *CovEC*. This corpus contains approximately 204 million words with more than 10 million unique words.

Table 3.4: Statistics of *CovEC*)

Corpus	#Words	#Unique Words	Max. Frequency
EC [2]	200,081,093	10,067,699	11,737,198
EC-BCovC (training set)	4,199,410	180,824	47,950
Total (in *CovEC*)	204,280,503	10,248,523	11,785,148

Table 3.4 highlights a key characteristic of Bengali – its inflected nature, where a significant portion of the most frequently used words are derived from conjunction structures. Bengali is known for its extensive verb and noun inflections, which contribute to the prevalence of conjunctions in the language. For instance, single conjunctions (e.g., the Bengali to English translation word: "oh") account for 5.76% of the entire embedding corpus, *CovEC*.

English Covid Embedding Corpus (ECoVE): The ECoVE corpus is an unlabelled collection of English texts exclusively related to Covid-19. These texts have been sourced from various online newspapers and social media platforms. To facilitate web data crawling, a scraper system has been created using the Scrapy library [110]. The data collection period spans from July 20, 2020, to June 19, 2021. An overview of the ECoVE corpus statistics is presented in Table 3.5. The

Table 3.5: Statistical summary of ECoVE

Attributes	Value
Total No. of text files	13,470
Total No. of sentences	92,870
Total No. of words	5,149,297
Maximum word frequency	249,685
Total No. of unique words	107,901
Total No. of unique words with min. frequency 2	59,820

ECoVE corpus consists of a total of 514,9297 words. Notably, the most frequently occurring single word appears 249,685 times, accounting for approximately 4.85% of the total word count. To enhance the quality of the data, English stop words have been excluded, resulting in a vocabulary of 107,901 words that have been incorporated into the dGloVe embedding model.

Arabic Embedding Corpu (AraEC): Table 3.15 presents the statistical details of the Arabic embedding corpus (AraEC). In response to the limited availability of pre-trained non-contextual embedding models, we compiled the AraEC by consolidating two separate corpora: the publicly accessible *SANAD* corpus [112], which contributed 194,797 text files, and AraEC itself, featuring an additional 12,000 texts that were randomly selected from the pre-processed text sets. Consequently, the combined AraEC comprises a total of 206,797 text files.

Table 3.6: Arabic COVID-19 embedding corpus (*AraEC*) statistics

Attributes	Values
No. of unlabelled texts	206,797
No. of sentences	2,481,564
No. of words	29,778,768
Maximum word frequency	637,198
No of unique words	767,699
Maximum no. of sentences	45
Minimum no. of sentences	1
Maximum text size	6.89 KB
Minimum text size	0.032 KB

The AraEC comprises a maximum of 767,699 unique words and 2,481,564 sentences. This corpus is utilized for generating non-contextual embedding models such as GloVe, FastText, and Word2Vec.

(ii) Text Classification Corpora Statistics

Bengali Authorship Classification Corpus (BACC-18): The BACC-18 (Bengali Author Classification Corpus-18) consists of the literary works of 18 renowned Bengali authors. This corpus was compiled by gathering texts from four online sources: NLTR (Society for Natural Language Technology Research) [113], Ebanglalibrary [114], Git repositories [106], and various blogs [107, 108, 109]. The largest number of texts (13,308) was extracted from NLTR, while the smallest set (240) was collected from blogs. The statistical summary of BACC-18 is presented in Table 3.7.

Bengali Text Classification Corpus (BDTCC): The developed corpus was partitioned into three sets where $106,899$ documents are used in training, $27,831$ documents in validation, and $21,477$ documents in testing sets, respectively. Table 3.8 shows the categorical summary. The highest number of text sets belongs to politics, whereas the lowest number of documents belongs to the environment

Table 3.7: BACC-18 Corpus Statistics

Author Name	No. of Testing text	No. of Training text
Bankim Chandra Chattopadhyay	179	927
Emon Jubayer	76	427
Humayun Azad	20	361
Kazi Nazrul Islam	84	390
Rabindranath Tagore	720	4438
Samaresh Majumdar	34	788
Sharadindu Bandyopadhyay	57	1705
Sarat Chandra Chattopadhyay	461	2539
Satyajit Ray	47	1960
Shamsur Rahman	196	1097
Sheteedu Mukherjee	42	1905
Sukumar Ray	61	425
Sunil Gangopadhyay	61	383
Swami Vivekananda	608	3666
Tareque Anu	22	674
Taslima Nasrin	58	370
Humayun Ahmed	97	618
Jibanananda Das	32	221
Total	2,855	22,894

Table 3.8: Categorical distribution of ($BDTC$)

Document category	No. of Training documents	No. of Validation documents	No. of Testing documents
sports	13,499	2,761	1,958
lifestyle	2,615	750	2,017
art	2,320	720	236
science & technology	3,554	1,200	1,183
education	6,590	2,300	970
environment	2,265	900	534
entertainment	11,194	3,520	3,439
economics	6,100	2,150	953
health	3,500	650	859
opinion	8,754	1,780	1,985
politics	25,490	5,600	3,713
crime	13,577	3,200	2,814
accident	7,441	2,300	816
Total	106,899	27,831	21,477

category. A random selection of 68.43% documents was allocated for the training set, 31.57% for the validation set, and 13.75% for the test set, respectively.

The difference in experiences, the annotation task itself, and the focus on the annotators contribute to a disagreement between them. Thus, it is required to

find how much the annotators agree in assigning text classes by using Cohen's kappa [111]. Cohen's kappa measures the agreement among annotators, and this determines how well one annotator agrees with another annotator. Thus, to investigate the standard inter-annotator agreement, a pairwise kappa coefficient is calculated. The developed corpus ($BDTC$) achieved a Kappa score (K) is 67.23%, which indicates a reasonable agreement among annotators.

Arabic COVID-19 Text Classification Corpus (AraCoV): The *AraCoV* corpus, outlined in Table 3.9, plays a pivotal role in Arabic COVID-19 text classification. It provides an extensive dataset comprising over 2.8 million words. Notably, the corpus maintains a balanced distribution with 9,099 COVID-related samples and 12,195 non-COVID samples, ensuring reliable classification outcomes. With a rich vocabulary encompassing more than 170,000 unique words in the non-COVID category and 55,000 in the COVID category, this corpus reflects the linguistic diversity in Arabic COVID-19 discussions. It further supports robust model evaluation by including separate training and testing subsets. In summary, the *AraCoV* corpus serves as a valuable resource, facilitating the development of precise and effective Arabic COVID-19 text classification models.

Table 3.9: *AraCoV* Corpus Statistics

Category	Attribute	Value
non-Covid (NC)	No. of words	2,269,388
	No. of unique words	170,120
	No. of samples	12,195
	No. of training/testing samples	8,537 / 3,658
	No. of sentences	49,657
Covid (CO)	No. of words	573,286
	No. of unique words	55,180
	No. of samples	9,099
	No. of training/testing samples	6,369 / 2,730
	No. of sentences	30,594

Because the corpus was collected between July 19, 2020, and March 20, 2022, all the newspapers predominantly covered similar types of news. Consequently, the total count of unique words is lower than the overall word count.

Bengali Covid Text Classification Corpus (BCovC): Table 3.10 provides statistics for the Bengali Covid text identification corpus ($BCovC$). The COVID category consists of sentences with a maximum of 20 words, while the non-Covid category has a maximum of 23 words. The minimum sentence length in both categories is 4 words. Despite having a total of 140,745 non-COVID samples, we

randomly selected 14,773 texts (approximately 10.5%) to address issues related to category-wise text sample imbalance[115]. Table 3.10 summarizes key statistics

Table 3.10: *BCovC* Corpus Statistics

Category	Attribute	Value
non-Covid	No. of words	2,866,371
	No. of unique words	122,241
	No. of samples	14,773
	No. of training/testing samples	10,331 / 4,442
	No. of sentences	318,485
Covid	No. of words	3,145,097
	No. of unique words	91,191
	No. of samples	14,340
	No. of training/testing samples	9,941 / 4,399
	No. of sentences	262,091

for the *BCovC* corpus, segregating data into non-Covid and Covid categories. In the non-Covid category, the corpus comprises 2,866,371 words with 122,241 unique words, 14,773 samples (divided into 10,331 for training and 4,442 for testing), and 318,485 sentences. In the Covid category, there are 3,145,097 words, 91,191 unique words, 14,340 samples (9,941 for training and 4,399 for testing), and 262,091 sentences. These statistics provide a comprehensive overview of the corpus's composition, aiding precise analysis and model development for COVID-19 text identification.

English Covid Text Classification Corpus (ECoVC): The English Covid-19 text mining corpus is a binary category labeled corpus, comprising Covid-19-related texts (CovidT) in one category and non-Covid-19-related texts (nCovidT) in another. An overview of the corpus is provided in Table 3.11.

Bengali Multi-Modal Text Classification Corpus (BMMTC): The Bengali Multi-Modal Text Classification Corpus (BMMTC) is a comprehensive dataset designed for multi-modal text classification tasks in the Bengali language. It combines textual and visual data, making it valuable for research in various applications, including natural language processing and computer vision. BMMTC enables the development and evaluation of models that integrate textual and visual information for improved classification accuracy. The statistical summary is presented in Table 3.12

Table 3.12 presents the statistical characteristics of the Bengali Multi-Modal Text Classification Corpus (BMMTC) categorized into Health, Sports, and Crime. The Health category consists of 2,500 multi-modal samples, with a word count of

Table 3.11: ECoVC Corpus Statistics

English Covid-19 Text Identification Corpus	
Category	**Description**
Type	Binary category labeled corpus
Categories	CovidT (Covid-19-related texts) and nCovidT (non-Covid-19-related texts)
Data Source	Online newspapers
Data Collection Period	July 1, 2021, to June 29, 2022
Data Crawling Policy	Scrapped based on *robot.txt* policy
Total Texts Scrapped	172,690 (named ECoVC)
Preprocessing	Applied to remove noisy and irrelevant information
Texts Removed	6,731 texts (3.90% of ECoVC)
Annotation Process	165,959 preprocessed texts manually annotated
Categorization	4,469 texts in CovidT category and 161,490 texts in nCovidT category
Addressing Imbalance	4,477 texts randomly selected for nCovidT category
Final Corpus	ECovC with 8,946 texts
Training-Testing Split	ECoVC randomly split into a 70:30 training and testing ratio
Annotation Reliability	Kappa value of 0.71, acceptable for downstream tasks

Table 3.12: BMMTC Corpus Statistics

Category	Attribute	Value
Health	No. of words	22870
	No. of unique words	18970
	No. of samples (Image&Texts)	2500
	No. of training/testing samples (Image&Texts)	2000 / 500
	No. of sentences	2500
Sports	No. of words	43896
	No. of unique words	36764
	No. of samples (Image&Texts)	4500
	No. of training/testing samples (Image&Texts)	3000 / 1500
	No. of sentences	4500
Crime	No. of words	31896
	No. of unique words	29348
	No. of samples (Image&Texts)	3500
	No. of training/testing samples (Image&Texts)	2500/ 1000
	No. of sentences	3500

22,870 and 18,970 unique words. In the Sports category, there are 4,500 samples, with 43,896 words and 36,764 unique words. The Crime category encompasses 3,500 samples, 31,896 words, and 29,348 unique words. Additionally, the ta-

ble provides the distribution of training and testing samples for each category. These statistics offer valuable insights into the composition and structure of the BMMTC corpus, facilitating research and analysis in multi-modal text classification tasks.

3.3 Active Learning-based Corpora Development

In response to **RQ1**, this study has devised an active learning-based corpus development system, representing a more dynamic and iterative approach to dataset creation. In active learning, an initial dataset is created with a limited number of labeled examples. Machine learning models are then employed to automatically select the most informative or uncertain data points for human annotation. Annotators label these selected instances, and the model is updated iteratively. The process continues, with the model becoming increasingly proficient at selecting the most valuable examples for annotation.

Developing a corpus for low-resource languages such as Bengali, Hindi, Italian, and others involves critical tasks like data collection, purification, and standardization. Past approaches typically relied on manual annotation and expert-level verification, leading to issues of time consumption, lower annotation quality, extensive human efforts, and high annotation costs. Furthermore, expanding manually annotated corpora is challenging and cumbersome, limiting their future usability. This study introduces an innovative ALSCD (Automated Language-Specific Corpus Development) framework tailored for low-resource languages, mitigating annotation ambiguity, reducing time, lowering corpus development costs, and enhancing future usability. Inspired by the MID4 dataset construction approach, the framework was applied to Bengali text corpus expansion. The ALSCD framework comprises six key components: Data Crawling, Preprocessing, Machine Annotation, Post Annotation, Verification, and Corpus Quality Measurement and statistics, as depicted in Fig. 3.2.

Data collection, purification, and standardization are important tasks in developing a corpus for low-resource languages like Bengali, Hindi, Italian, and others. Most past low-resource corpora have been developed based on manual annotation and expert-level verification schemes [116, 117]. The manual data labeling or annotation process suffers from a considerable amount of time, lower quality of annotation, and enormous human efforts with a high annotation cost. Moreover, the expansion of the manually annotated corpus is complicated and tedious, leading to decreased usability of the manually annotated corpora. This study introduces an *intelligent* ALSCD framework for corpus construction in low-

resource languages, which reduces the annotation ambiguity, annotation time, and corpus development cost and improves the future usability of the corpus. The proposed framework followed the philosophy of MID4 dataset construction research and experimented on Bengali text corpus accretion research [118]. Fig. 3.2 shows an abstract view of the proposed ALSCD framework, which consists of *six* significant constituents: Data Crawling, Preprocessing, Machine Annotation, Post Annotation, Verification, and Corpus Quality Measure & Statistic.

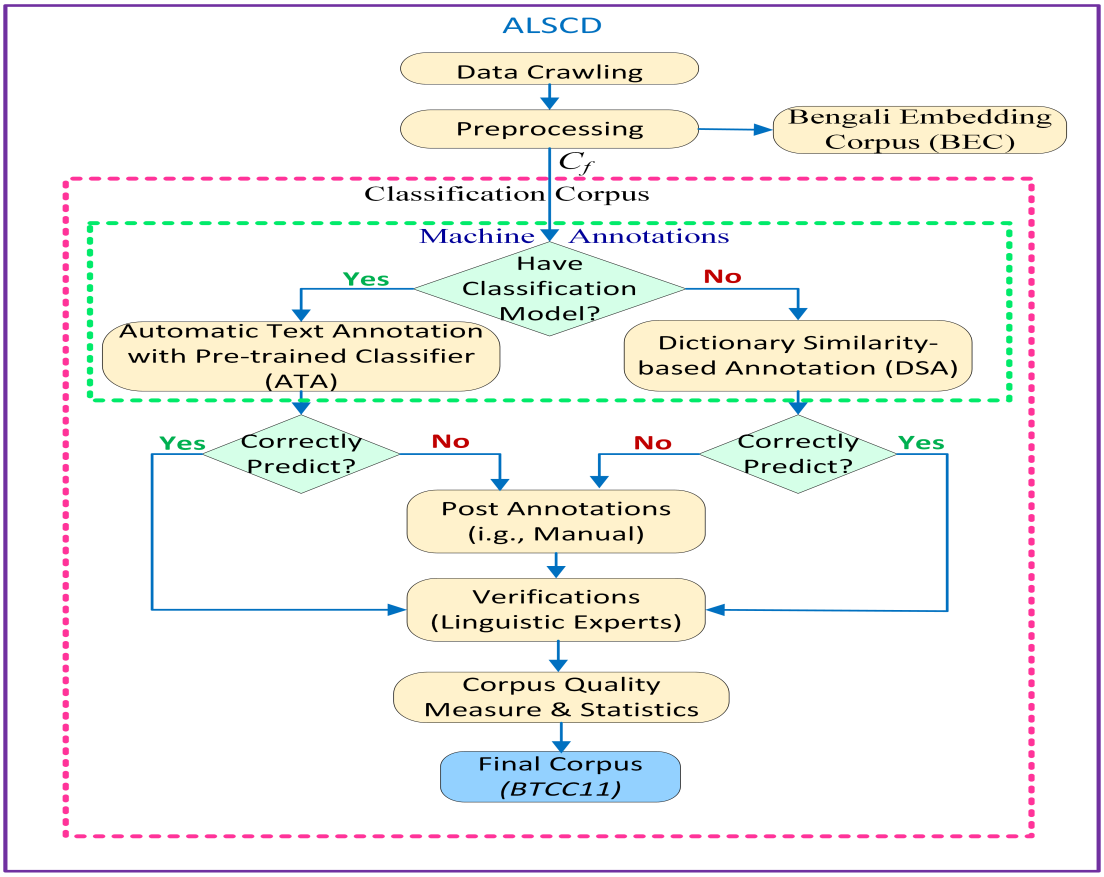


Figure 3.2: Abstract view of active learning-based semi-automated corpus development (ALSCD) framework

3.3.1 Data Crawling

The data crawler selects some Bengali web page URLs based on robot.txt policy, domain types, dates, geographical areas, trends, and popularity [101]. Data crawling uses a Python library named *Scrapy* [110]. Scrapy recursively scraped the abandoned structured data from the base URL to all possible content of URLs. The text in the HTML page is converted using the inscriptis Python library [119]. A total of 1,407,049 Bengali text files have crawled between two phases. In the first phase, 1,020,000 unlabelled text files have been collected from

January 2010 to December 2019 [2]. In the second phase, 387,049 unlabelled text files have been newly harvested from January 2020 to March 2021. Thus, a total of 1,407,049 crawled texts are saved in a ".txt" format with "UTF-8" encoding and sent to the preprocessing phase.

3.3.2 Data Preprocessing

The noisy text misleads retrieval of the actual information and degrades the performance of the downstream tasks. Thus, a noise reduction filter is used to remove all irrelevant constituents from the crawled texts. The noise reduction filter involves the following steps:

- Non-Bengali alphabets and digits are removed with a NULL value.
- All the punctuation from texts with a NULL value such as#$%&+ = {}[] : ‖' <> /; eliminated.
- Multiple newlines and white space are replaced by a single newline and white space.
- Hashtags, HTML tags, and URLs are substituted by a single space.
- All the duplicate texts are removed.
- A single-character word is replaced by a NULL value.

A total of 12,720 text files appeared blank due to the preprocessing operations and were removed from the corpus. Now, a total of 1,394,329 text files have been randomly partitioned into two parts: the first part randomly selected 19.00% text files (e.g., 264,922) for classification corpus ($BTCC11$), and the remaining (1,129,407) unlabelled files are directly used for embedding corpus (BEC).

3.3.3 Machine Annotation

The machine text annotation process takes the unlabelled text and generates the labeled and unlabelled texts based on a rank-based probability score. Fig. 3.2 illustrates two scenarios for low-resource languages: dictionary similarity-based annotation (DSA) and automatic text annotation with a pre-trained classifier(ATA).

3.3.3.1 Dictionary similarity-based annotation (DSA)

In this approach, an unlabelled text is fed to the module that produces a labeled or unlabelled text based on the probability value. Fig. 3.3, shows the schematic representation of the DSA scheme.

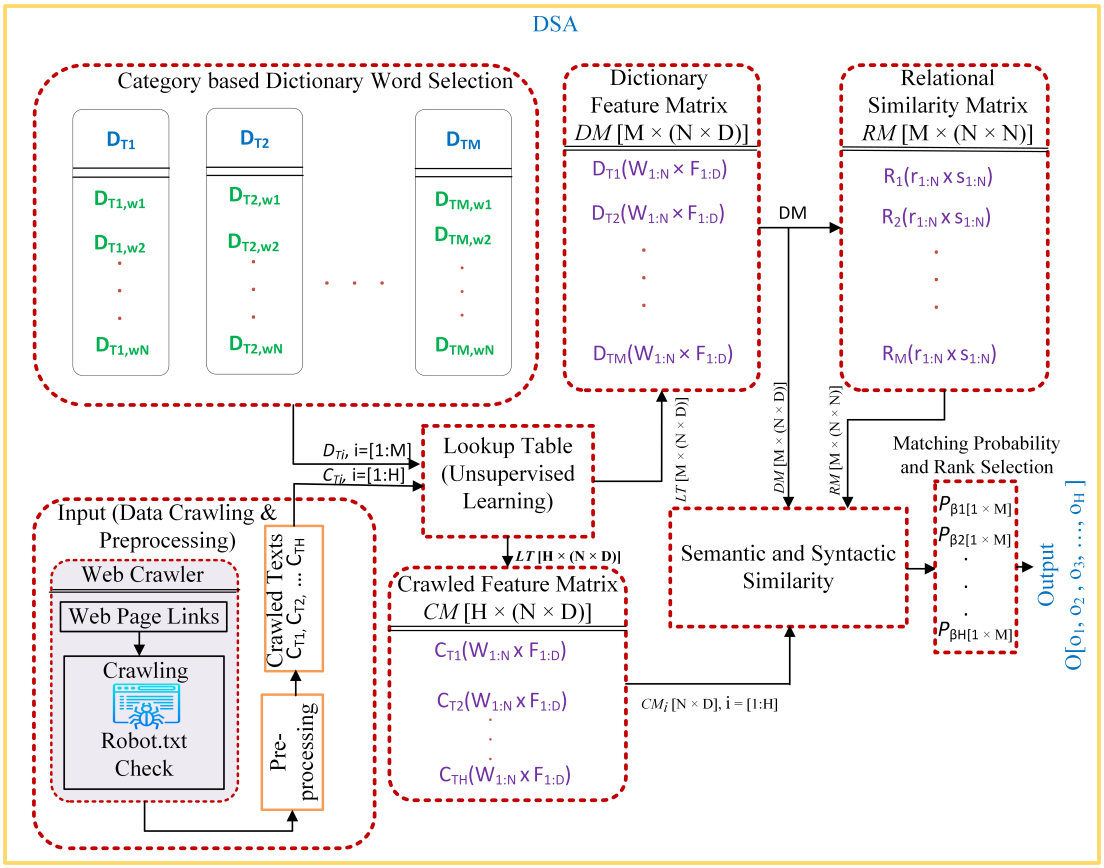


Figure 3.3: Schematic representation of dictionary similarity-based annotation (DSA) scheme

The annotation scheme consists of eight major phases: (i) category-based dictionary word selection, (ii) data crawling and preprocessing, (iii) lookup table (unsupervised learning), (iv) dictionary and crawled feature matrix generation, (v) relational similarity matrix generation, (vi) semantic and syntactic similarity, (vii) crawled text matching probability and rank selection (viii) output (i.e., category name) generation. The overall scheme, including all phases, is summarised in DSA Algorithm 1. This algorithm aims to maximize the annotation text with a minimal error rate.

Algorithms 2, 3 and 4 illustrates the sub-process of DSA Algorithm 1.

Category-based dictionary word selection: The proposed system selects the number of categories for the classification corpus. Suppose $\{D_{T1}, D_{T2}, D_{T3}, ..., D_{TM}\}$ is the set of dictionaries, where D_{TM} defines the T^{th} dictionary and M denotes

Algorithm 1: Dictionary Similarity-based Annotation (DSA) Algorithm

1 *step*1 : Select category based dictionary word
2 *step*2 : Crawled and preprocess the input data
3 *step*3 : Select word feature from lookup table
4 *step*4 : Produce feature matrix ($DM\&CM$)
5 *step*5 : Construct relational matrix (RM)
6 *step*6 : Match Semantic and syntactic similarity (Algorithm 2 & 3)
7 *step*7 : Produce crawled text matching probability and select matching rank (Algorithm 4)
8 *step*8 : Return Output. //labeled or unlabelled text

the total number of dictionary or category, which means each category represent a dictionary. Each of the dictionaries is representative of the corresponding category where N popular words are selected for the corresponding category. The number of words has an immense impact on DSA performance, which is shown in Sec. 3.4.1 (Fig. 3.4). The $[D_{Ti,w1}, D_{Ti,w2}, D_{Ti,w3}, ..., D_{Ti,wN}]$ is the vector representation of the i^{th} dictionary and $D_{Ti,wN}$ indicates the N^{th} word of the i^{th} dictionary. Finally, the dictionary and words are expressed as a 2D ($M \times N$) matrix. This dictionary is used for similarity checking in the next phases. In this research, $N = 200$ (number of words), and $M = 11$ (total number of categories). The categories are: *Accident (01)*, *Climate & Environment (02)*, *Crime (03)*, *Education (04)*, *Entertainment (05)*, *Health (06)*, *Law & Justice (07)*, *Lifestyle (08)*, *Politics (09)*, *Science-Tech (10)*, and *Sports (11)*. The dictionary matching probability value depends on the number of dictionary words, e.g., when the number of words is increased in the dictionary but the number of crawled text matching words is not increased, then the overall matching probability decreases. Therefore, the simulation achieves maximum dictionary matching probability at 200 words.

Input (Data crawling & preprocessing): The input data is prepared in two steps: (i) crawling and (ii) preprocessing (Fig. 3.3). A total of $1,407,049$ text files have been collected from the Web portal based on robot.txt policy [101]. This work randomly selects $264,922$ texts for dictionary similarity-based annotation. The vector $[C_{T1}, C_{T2}, C_{T3}, ..., C_{TH}]$ indicates the preprocessed crawled texts, where C_{TH} denotes the T^{th} text and H represents the total number of texts used in the annotation process. Each of the crawled text propagates through the subsequent phases and assigns an appropriate labeled. Sections 3.3.1 and 3.3.2 describe the data crawling and preprocessing steps in detail.

Lookup table (Unsupervised learning): The lookup table contains 300

features for each of the 2,867,701 words arranged in a 2D matrix (e.g., 2867701×300). The input words and dictionary-wise raw words similarity are not good for semantic and syntactic similarity. As a result, the lookup table contains a pre-trained word embedding model which is trained with Bengali embedding corpus (BEC) using unsupervised learning techniques (e.g., GloVe, FastText, Word2Vec & Meta-Embedding). The lookup table takes the i^{th} dictionary words D_{Ti} (e.g., 200 words) as an input and outputs a 2D (e.g., $N \times D$) dictionary feature matrix using a mapping function $M(D_{Ti}, ME_B[1:3])$. The lookup table also takes the i^{th} crawled text words C_{Ti} (e.g., truncated 200 words) as an input and outputs a 2D crawled feature matrix using a mapping function $M(C_{Ti}, ME_B[1:3])$. Here, N and D indicate the number of words and feature dimension, whereas $ME_B[1:3]$ represents the top-performing three ME models.

Dictionary and Crawled feature matrix generation: Two types of feature matrices are generated: one for dictionary words (e.g., dictionary feature matrix (DM)) and another for crawled text (e.g., crawled feature matrix (CM)). The dictionary and crawled feature matrices are generated using Eq. (3.2).

$$\begin{cases} DM[D_{Ti} \times (W_{1:N} \times F_{1:D})] = Mi(D_{Ti}, E_{ME}[1{:}3]), i=1,...,M \\ \\ CM[C_{Ti} \times (W_{1:N} \times F_{1:D})] = Mi(C_{Ti}, E_{ME}[1{:}3]), i=1,...,H \end{cases} \tag{3.2}$$

Here, M indicates the number of dictionaries and sequentially mapped the D_{Ti} to the function $Mi(.)$ which produces a $[W_{1:N} \times F_{1:D}]$ vector. Similarly, H denotes the total crawled texts. Finally, the dictionary and crawled feature matrices are stacked one over another to produce a 3D (e.g., $M \times (N \times D)$, $L \times (N \times D)$) representation.

Relational similarity matrix generation: The relational similarity matrix contains the cosine similarity score of N words of i^{th} dictionary. The i^{th} dictionary D_{Ti}, calculates the cosine similarity score of j^{th} word using Eq. 3.3.

$$RM[i : r_j : S_k] = C_{sim}([(D_{Ti}(W_j : F_{1:D}))], [(D_{Ti}(W_k : F_{1:D}))]), k = 1, ..., N \tag{3.3}$$

Here, C_{sim} denotes the cosine similarity function whereas $[(D_{Ti}(W_j : F_{1:D}))]$ represents j^{th} word feature vector of i^{th} dictionary . The $[(D_{Ti}(W_k : F_{1:D}))]$ indicates the i^{th} dictionary words iterate through the variable k, and $k = 1, 2, 3, ..., N$. In this way, i^{th} dictionary produces a $(N \times N)$ cosine similarity matrix and sorts the matrix in descending order (e.g., higher to lower). Finally, M dictionaries will produce a 3D similarity matrix (e.g., $RM[M \times (N \times N)]$). This matrix ($RM$) is used in the semantic and syntactic similarity phase to measure the inferring relation between the crawled text word and dictionary word [120]. The RM is

also utilized to measure the transitivity or triangular relation among the crawled word with the dictionary words [121]. This matrix also is used for optimal matching value selection purposes when the cosine similarity score of two words infers that the words are the same or different.

Semantic and syntactic similarity: Three matrices, such as crawled text feature matrix (CM), dictionary feature matrix (DM), and relational similarity matrix (RM), are taken as the input to produce a dictionary-wise probability matching matrix P_β (e.g., $(H \times M)$) using Eq. 3.4.

$$P_\beta[i] = dictionaryMatching(DM_j, CM_i, RM_j), i = 1, 2, 3, ..., H; j = 1, 2, 3, ..., M \tag{3.4}$$

Each crawled text represents a $(N \times D)$ dimension feature matrix. This feature matrix calculates the word-wise cosine similarity values. The matching of the i^{th} crawled text of k^{th} word feature α_1 with the j^{th} dictionary k^{th} word feature α_2, depends on the following two rules (R1 & R2):

- R1: If similarity value γ is greater than the first similarity value (T_{mx}) then α_1 is directly inferring to α_2
- R2: If γ is smaller than T_{mx} and greater than T_{mn} and relational similarity value ν is greater than T_{mx} then α_1 is transitive closer of α_2.

The first maximum (T_{mx}) and the second maximum (T_{mn}) similarity scores are selected with an experimental Algorithm 2. The experimental method simulates the similarity score from 30.00 to 70.00. It selects a T_{mx} and a T_{mn} such that the error between expert feedback and cosine similarity output of two words is minimal. Algorithm 2 shows the step-by-step simulations.

Sec. 3.4.1 describes a detailed impact of T_{mx} and T_{mn} on dictionary matching probability [Fig 6.8]. The simulations were conducted on dictionary words, whereas word-word similarity scores were taken from the relational matrix (RM). In this experiment, the best performing T_{mx} and T_{mn} are 65.00 and 45.00, respectively. The matching probability for each of the dictionaries for i^{th} crawled text words is calculated using the fraction $\frac{M_{ct}}{N}$. Here M_{ct} denotes the total number of words in i^{th} crawled text matching with j^{th} dictionary, and N stands for the total number of words in the text and dictionary. The 2D probability matrix P_β contains M dictionaries probability values for each of the crawled text. The dictionary probability is not disjoint. Algorithm 3 shows the procedure for $dictionaryMatching$ function (Eq. 3.4) which calculates the probability matrix P_β. The matrix P_β is used for assigning a category name of a crawled text based on rank-based probability matching score [122].

Algorithm 2: First and Second Maximum Word Similarity (FSMWS) score selection Algorithm

Function : $FSMWS(S_W, DM, RM, M, N)$ // Input Function {
$T_{mx} := 0, T_{mn} := 100, \Delta_{mn} := \infty$ //Initialization
$Mx := 70, Mn := 30$ //Max and Min word-word similarity score
for $i \leftarrow Mn \rightarrow Mx$ **do**
  for $j \leftarrow Mn, t_s := 0 \rightarrow Mx$ **do**
    for $k \leftarrow 1 \rightarrow M$ **do**
      $\Delta_{er} := \sum_{l=1}^{N} Sim(cosine(DM[k:l], S_W[l]), RM[k:l], i, j)$
      $t_s = t_s + \Delta_{er}$
    end
    if $t_s < \Delta_{mn}$ **then**
      $\Delta_{mn} := t_s, T_{mx} := i, T_{mn} := j$
  end
end
return T_{mx}, T_{mn} //Output}

Algorithm 3: Similarity measure of CM with DM (SMCF)

Function : $dictionaryMatching(H, M, N, DM, CM, RM)$ //Input Function {
$T_{mx} := 65.00$, $T_{mn} := 45.00$ //1st & 2nd max similarity score
$P_\beta := [[0] * M] * H$
for $i \leftarrow 1 \rightarrow H$ **do**
  $\vartheta := []$
  for $j \leftarrow 1 \rightarrow M$ **do**
    $M_{ct} := 0$
    for $k \leftarrow 1 \rightarrow N$ **do**
      $\alpha_1 := C_{Ti}(W_k : F_{1:D})$
      $\alpha_2 := D_{Tj}(W_k : F_{1:D})$
      $\gamma := C_{sim}(\alpha_1, \alpha_2)$
      $\nu := R_i(r_j : S_{k+1})$
      if $\gamma \geq T_{mx}$ **then**
        $M_{ct} := M_{ct} + 1$
      if $\gamma \geq T_{mn}$ *&* $\nu \geq T_{mx}$ *&* $\gamma < T_{mx}$ **then**
        $M_{ct} := M_{ct} + 1$
    end
    $\vartheta[j].append(\delta)$, $\delta := \frac{M_{ct}}{N}$
  end
  $P_\beta.insert(i, \vartheta)$
end
return (P_β) // Output }

Matching probability and rank selection: This module takes the matching probability matrix P_β as an input and outputs the category of texts as a 1D vector $O = [o_i | i = 1...H]$. Here o_i denotes the i^{th} text output defined by $o_i \in \{labelled, unlabelled\}$. The labeled or unlabelled text tag depends on rank-based probability matching score [123]. The rank-based probability matching score is obtained from an empirical evaluation using Algorithm 4. Algorithm 4 minimizes the gap between human level annotation (δ_{l_h}), and machine level annotation (δ_{l_m}). Rank r is defined by the amount of texts for which the gap between δ_{l_h} and δ_{l_m} is minimum.

Algorithm 4: Optimal Rank Selection (ORS) based on probability matching score

$Function : ORS(P_\beta)$ // Input Function {
$S_f := df.P_\beta(frac = 1)$
$\nabla_t := 0.75$
$D_s := np.array_split(S_f, 50)$
$R_{idx:=randint(1,50)}$
$\widehat{D_s} := D_s[R_{idx}]$
$\delta_{l_h} := \eta_h(\widehat{D_s})$
$\tau = len(\widehat{D_s})$
$\beta_t := 0$
$\Delta_m := inf$
for $r \leftarrow 50 \rightarrow 80$ **do**
    $\delta_{l_m} := \eta_m(\widehat{D_s}, r)$
    $\Delta_e := \sum_{i=1}^{\tau} fabs(\delta_{l_h} - \delta_{l_m})$
    if $\Delta_e \leq \Delta_m$ *or* $fabs(\Delta_e - \Delta_m \leq \nabla_t)$ **then**
        $\Delta_m := \Delta_e$
        $\beta_t := r$
end
$return(\beta_t)$ // Output: optimal rank }

A small chunk of data $\widehat{D_s}$ (about 2%, 5298 texts) is randomly selected for the experiment. The expert manually verifies $\widehat{D_s}$. The machine-level annotation probability of $\widehat{D_s}$ and rank value r are mapped using function $\eta_m(\widehat{D_s}, r)$ which returns label δ_{l_m}, i.e., $\delta_{l_m} = \eta_m(\widehat{D_s}, r)$. Rank 1 means 1% data are selected from $(\widehat{D_s})$. The rank value starts from 50 (e.g., top 50% data are selected from $\widehat{D_s}$) and goes up to 80, incremented by 1. Function η_m finds the number of maximum matching probabilities within the rank r. The texts with maximum matching probabilities are considered correct machine-level annotations, and the remaining texts are considered unlabelled. The annotation error difference between machine and human level is denoted as Δ_m and initialized with a maximum value. Over

iterations, Δ_m is to be minimized, and rank value r is to be maximized until tolerance ∇_t is reached. This parameter ∇_t maximizes the machine-level annotations. Finally, β_t is the optimal rank r for chunk $\widehat{D_s}$. The remaining corpus texts are labeled based on the optimal rank value r (e.g., r = β_t = 72). The impact of rank value on DSA performance is presented in Sec. 3.4.1 [Fig. 3.6].

Output (category name): The dictionary similarity-based annotation generates the output as $O = [o_i|i = 1...H]$ and each of the output has a tag (e.g., labeled or unlabelled) assigned based on the optimal rank value (β_t). If the i^{th} label cannot annotate with the machine-level annotation method, then it is transferred to the post-annotation phase; otherwise, it is transferred to the verification phase. The DSA method has annotated 191, 719 texts out of 264, 922, and the remaining 73, 203 texts are transferred to the post-annotation and verification phases. In the verification phase, the linguistic expert verifies the machine-level annotation.

3.3.3.2 Automatic text annotation with pre-trained classifiers (ATA)

This module uses active learning techniques for expanding the classification corpus. At the initial trial, a small amount of labeled text is added to the classification corpus (e.g., 186,474) using the DSA method. A classification framework (ME-CNN) is built for the classification model and is used in the subsequent trial. Out of 82,620 texts, 76,010 are annotated with the ATA method, 6,610 texts are transferred to the post-annotation phase, and 76,010 are distributed to the verification phase. However, if the classification framework starts with a small amount of labeled data, the classification model performance is not good enough. For this reason, this research includes an active learning method, which improves the classification quality for low-resource languages. Section 3.4.1 describes the impact of active learning.

3.3.4 Post Annotations

According to the first trial, the DSA method annotates the unlabelled texts to labeled texts. In this paradigm, 264,922 text files are input to the DSA method, annotating 191,719 unlabelled texts. A total of 191,719 texts are moved to the verification phase. The post-annotation phase performs annotation of the remaining 73,203 unlabelled texts. Two human annotators conduct the post-annotation process. An empirical rule is used for the post annotation, e.g., *if a text containing more words is semantically and syntactically related to a predefined category, then the text is assigned a tag from the predefined category.* This research considers 11

predefined text classes: Accident (01), Climate & Environment (02), Crime (03), Education (04), Entertainment (05), Health (06), Law & Justice (07), Lifestyle (08), Politics (09), Science-Tech (10) and Sports (11). If a text tag does not belong to one of the classes, then it is discarded from the corpus. The ATA method annotated 76,010 texts, which were transferred to the verification phase. Among 73,203 texts, both annotators agreed on 49,046 texts, disagreed on 17,716 texts, and had different opinions on 6,441 texts. Finally, 49,046 texts are included in the final corpus ($BTCC11$), 6,441 are sent to the verification phase, and 7,716 texts are discarded. Similarly, the second trial (of the ATA method) takes 82,620 unlabelled texts, where 76,010 texts are labeled by the ATA method. The remaining 6,610 texts are sent to the post-annotation phase. The post annotators agreed on 5,816 texts, disagreed on 538 texts, and the rest of the 256 texts caused confusion with the annotators, which moved through the verification phase. A total of 5,816 texts have been added to the $BTCC11$, whereas 538 texts are discarded from the corpus.

3.3.5 Verification

In the first trial, 191,719 texts were annotated using the DSA method, and 6441 were obtained from the post-annotation phase for verification. The entire labeled text is verified by a linguistics expert. Out of 191,719 labeled texts, 132,920 labeled are agreed upon by the DSA method and expert. A total of 58,799 texts are eliminated from the corpus due to disagreement between the method and the expert. Regarding post annotation, the expert agreed on 2,368 (first annotator) and 2,140 (second annotator) labels. The expert has added 69.99% of rejected texts from the post-annotation phase during the verification process. In the first trial, a total of 186,474 (137,428 from verification & 49,046 from post annotation) texts are included in $BTCC11$. A total of 78,448 (e.g., 60,732 from verification & 17,716 from post-annotation) texts are discarded from the corpus due to disagreement. In the second trial, the verification phase received 76,010 texts from the ATA process, and 256 texts were taken from the post-annotation phase. Due to the disagreement between the expert and ATA, about 14,438 texts were excluded from the corpus. At the same time, 189 texts are included from the post-annotation phase, whereas 67 are discarded from the corpus. Finally, 67,577 texts are included in the $BTCC11$ (61,761 for verification & 5,816 for post annotation). Overall, 73.82% of text documents are added due to expert verification.

3.4 Evaluation of the Corpora

Various qualitative and quantitative measures can be taken to assess the reliability or quality of a corpus [124, 125]. In the following subsections, we present the qualitative and quantitative measures of the developed ALSCD framework for ATA with pre-trained classifier and DSA Algorithm performance.

3.4.1 Qualitative Measures

The performance of the proposed ALSCD framework depends on two processes: ATA and DSA. Kappa score, accuracy, and error measures are used to assess the performance. The *accuracy* is measured by the term $(\frac{e_a}{m_a} \times 100)$, here m_a denotes the number of text annotated by the method (e.g., DSA or ATA) and e_a represents the total number of texts approved for correctly labeled by the expert. The error is calculated by the expression $(100-(\frac{e_a}{m_a} \times 100))$. The average accuracy of DSA and ATA is the accuracy of the ALSCD framework.

Kappa score (κ): Cohen's kappa (κ) is used to calculate the inter-annotator reliability score [111]. Kappa score calculates the annotation quality based on the texts considered in the machine-level annotation. If any disagreement occurs between the annotators, it is resolved by the linguistics expert in the verification phase. Table 3.13 shows the class-wise kappa score.

Table 3.13: First & Second trail inter-annotator agreements with class wise kappa statistics

	First trial					Second trial				
Class code	$a_1 \wedge a_2$	$d_1 \wedge d_2$	d_1	d_2	κ_1	$a_1 \wedge a_2$	$d_1 \wedge d_2$	d_1	d_2	κ_2
01	3737	1720	170	163	0.86	310	46	07	05	0.87
02	1354	170	79	109	0.58	400	44	20	15	0.67
03	3917	1148	377	350	0.67	170	38	11	14	0.68
04	4189	1103	259	269	0.75	230	37	07	12	0.76
05	5108	2031	280	197	0.85	574	45	06	09	0.84
06	1780	197	113	102	0.59	676	46	16	19	0.70
07	4780	2150	409	326	0.78	750	56	17	06	0.81
08	4909	2029	420	310	0.77	611	55	12	11	0.80
09	4816	2408	370	357	0.79	709	54	08	14	0.82
10	6570	2410	376	405	0.80	680	53	09	15	0.80
11	7886	2350	530	470	0.76	816	64	10	13	0.83
Total	49046	17716	3383	3058	8.2	5926	538	123	133	8.58
Avg.	$\overline{\kappa_1} = 0.74$					$\overline{\kappa_2} = 0.78$				

In the first trial post-annotation phase, the maximum degree of agreements was achieved (e.g., κ value) for class 01 (0.86), whereas the minimum agreements

occurred for class 02 (0.58). Class 02 and Class 06 have some common words or mutual word semantics, which results in low kappa values and causes an ambiguous situation for the annotators. The first trial post-annotation phase has an average kappa ($\overline{\kappa_1}$) value of 0.74, which indicates the substantial agreement between annotators. The maximum reliability score is 0.87 (Class 01) in the second trial, and an average kappa ($\overline{\kappa_2}$) score is 0.78. This score indicates a substantial agreement between annotators.

DSA performance: The DSA method is used in the first trial. The performance of this method relies intensely on the number of words in the dictionary of each category. Fig. 3.4, shows the impact of the number of words per dictionary on accuracy. The maximum accuracy (e.g., 69.33%) is achieved for 200 words per dictionary. The DSA performance increases from 25 to 200 words and then falls. Therefore, the number of words per dictionary is chosen to be 200 for the DSA algorithm.

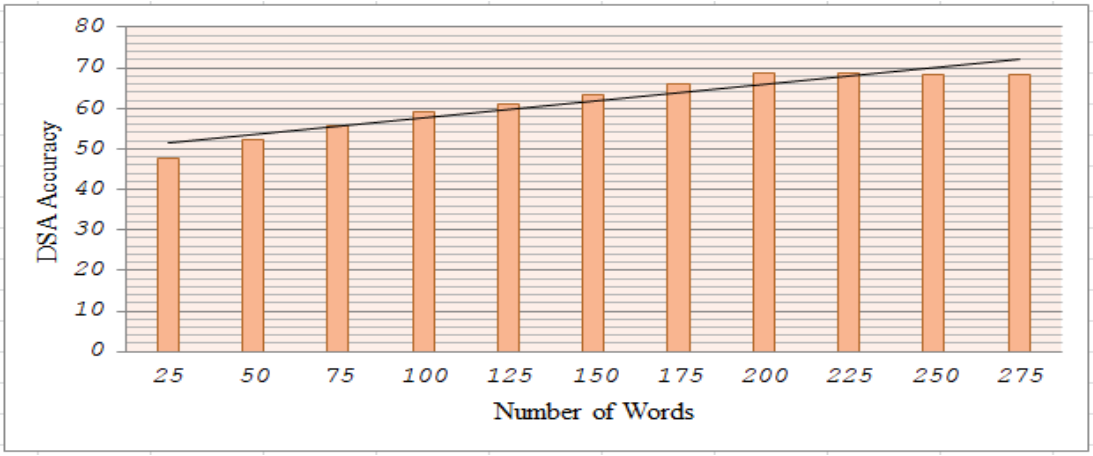


Figure 3.4: Impact of number of dictionary words on DSA Accuracy

The word-word cosine similarity score refers to the semantic or syntactic similarity between two words. Two words are considered similar if and only if the word-word similarity value satisfies constrained: T_{mx} and T_{mn}. The maximum matching probability of the i^{th} crawled text (e.g., C_{Ti}) with the categorical dictionary depends on the number of words matching with dictionary words. Therefore, the dictionary matching probability increases when more crawled words are matched with dictionary words. Thus, the DSA Algorithm annotates the texts with a higher matching probability [126]. Fig. 6.8 shows the different combinations of the first and second maximum values (i.e., T_{mn} & T_{mx}) with the average dictionary matching probability.

Fig. 6.8 illustrates that the maximum average dictionary matching probability is 23.9% when $T_{mn} = 45$ and $T_{mx} = 65$. Fig. 3.6 shows the impact of the rank

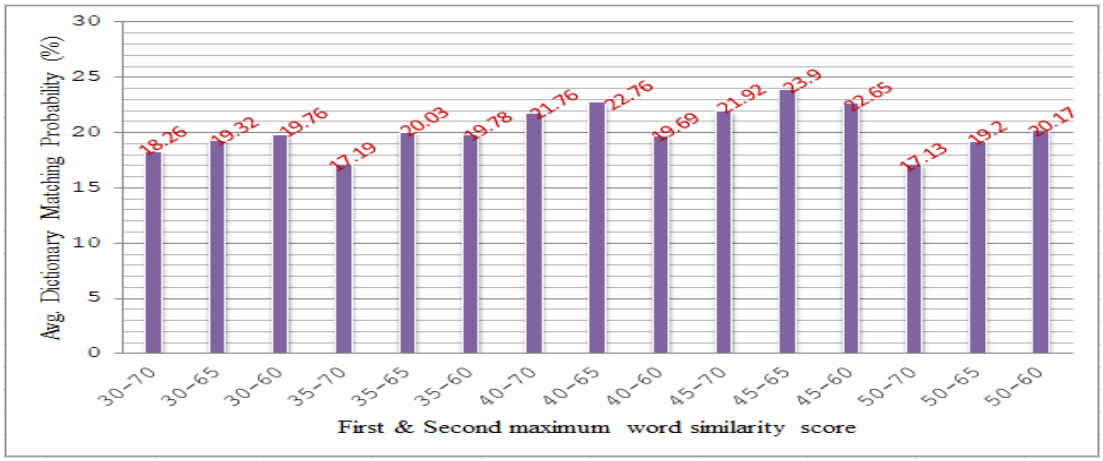


Figure 3.5: Average dictionary matching probability vs. second (T_{mn}) and first (T_{mx}) maximum word similarity scores

value on the number of annotated texts and DSA error computed by the DSA algorithm. DSA error is the number of incorrect annotated texts. According to Algorithm 4, the rank value is defined as the percentage of text selected by the DSA technique. For example, R50 indicates 50% (132,461 out of 264,922) texts are annotated by the DSA algorithm based on probability matching score.

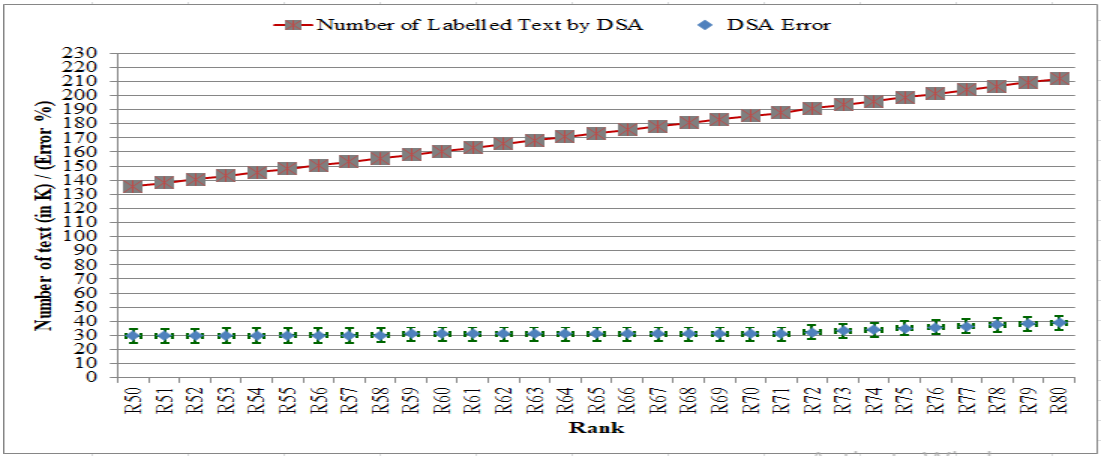


Figure 3.6: Text annotation capability at the rank value of DSA algorithm

The DSA algorithm annotates about 132K text with a minimum error of 29.83% at R50. At the same time, the maximum error is 39.00% at R80 for 200K text. Based on Algorithm 4, the DSA Algorithm tries to maximize the annotation number with the global minimum and consecutive rank errors. The proposed algorithm obtained the best performance at rank 72 (e.g., R72) with 30.67% error and 191,719 annotated texts.

ATA performance: The ATA method is used in the second trial. In the first trial, 186,474 texts are labeled by the DSA algorithm and post-annotation phase.

Among these, 105,543 texts are used to generate the ATA model. A single-layer multi-kernel CNN model is used for the ATA model [101]. The classifier model tries to classify a total of 82,620 unlabelled texts into pre-defined classes. Based on the softmax confident value, 76,010 texts were annotated by the ATA method, whereas it failed to annotate 6,610 texts. The expert verified a total of 61,572 texts (out of 76,010). Thus, in the second trial, the ATA method achieved an accuracy of 81.00%.

ALSCD framework performance: The proposed ALSCD framework is annotated 254,051 texts using the DSA and ATA methods. In the first trial, the DSA method correctly annotated 132,920 texts (out of 191,719) with 69.33% accuracy. In the second trial, 61,572 texts were annotated by the ATA method with 81% accuracy. The post-annotation and verification phases annotate the remaining 59,559 texts. The ALSCD framework achieved an average accuracy of 75.16% for automatic annotation (69.33% by DSA & 81% by ATA).

3.4.2 Quantitative Measures

The developed text corpus can be assessed by several quantitative measures such as the total number of sentences, unique words, sentence length, size, word frequency, and train-test set distribution [127]. ALSCD developed two corpora: text classification ($BTCC11$) and embedding (BEC). A total of 1,394,329 usable texts have crawled from four different sources: newspapers, literature, social media, and blogs. Fig. 3.7 shows the domain-wise data distributions. A maximum of 90.00% of data is collected from online newspapers, and a minimum of 2.00% of data is accumulated from blogs. All the data are crawled based on robot.txt policy [101]. A total of 18.00% texts are selected randomly for preparing the classifiers, and the remaining texts are used for embedding. Table 3.14

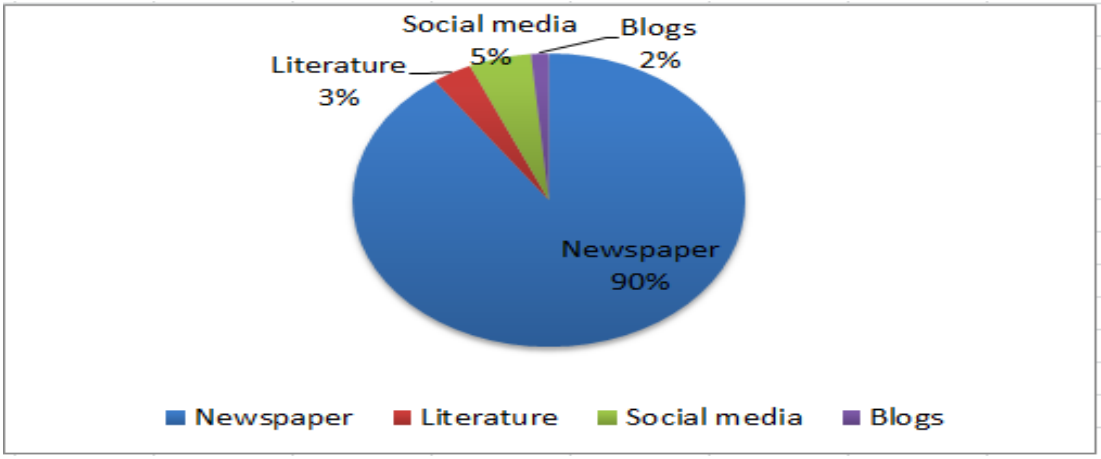


Figure 3.7: Domain-wise distribution of crawled texts

shows the class-wise training and testing texts distribution of the classification corpus ($BTCC11$) in the first and second trials.

Table 3.14: Class-wise distribution of the classification corpus ($BTCC11$)

	First trial		Second trial
Class Code	No. of train text	No. of test text	No. of train text
01	4251	4503	8662
02	1276	3204	3789
03	8358	6213	7958
04	9567	6783	6504
05	16466	11709	2180
06	1869	3352	6397
07	13159	7525	5426
08	11532	5993	7589
09	8069	8260	4896
10	10178	14287	9103
11	20818	9102	5073
Total	105543	80931	67577

In the first trial, 105,543 training and 80,931 testing texts were accumulated using the DSA algorithm. A total of 67,577 texts have been annotated in the second trial using the ATA method. All labeled texts are included in the training set of $BTCC11$. The variation of texts in each class occurred due to text availability on the web. Table 3.15 illustrates statistics of embedding corpus (BEC). To the best of our knowledge, this is the largest Bengali unlabelled corpus in terms of maximum unique words (e.g., 10.86 M) and domain diversity. The corpus contains a total of 1.14 million (M) unlabelled texts with 364.05 M different sentences. The maximum text size of the document is 107.884 KB, whereas the minimum size is 0.048 KB.

Table 3.15: Bengali embedding corpus (BEC) statistics

Attributes	Values
No of unlabelled texts	1,129,407 (1.14 M)
No. of sentences	364,055,081 (364.05 M)
No. of words	1,146,833,902 (1146.83 M)
Maximum word frequency	11,737,198 (11.73 M)
No of unique words	10,867,699 (10.86 M)
No. of unique words with min. frequency 2	2,867,701 (2.86 M)
Maximum no. of sentences	376
Minimum no. of sentences	1
Maximum text size	107.884 KB
Minimum text size	0.048 KB

Fig. 6.6 shows the variation of text sizes with different word frequencies in BEC. The text size of 1-1.99 KB carried the maximum word frequency (228,597),

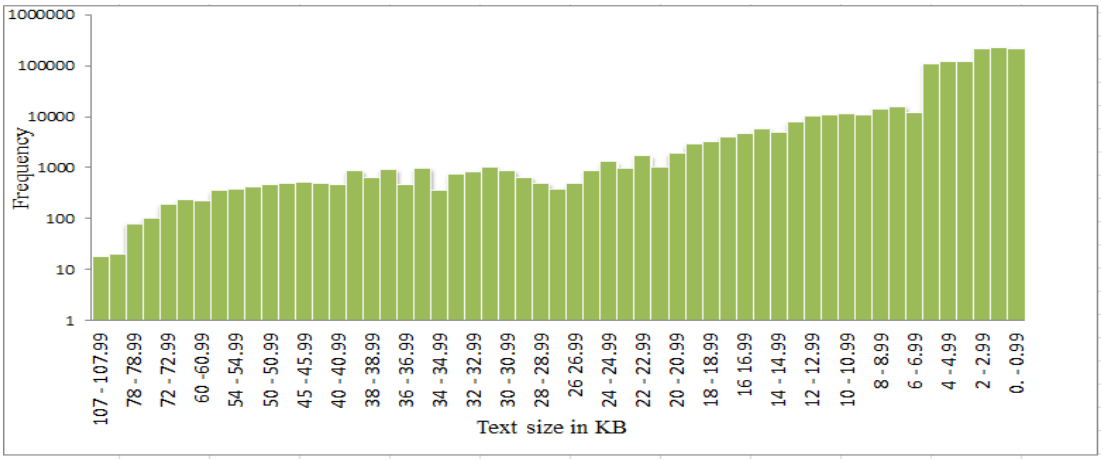


Figure 3.8: Variations of text size (in KB) with different word frequencies of BEC

whereas the minimum frequency of 18 is found for the text size between 107 to 107.99 KB. It is observed that the amount of text decreases with increasing frequency.

3.4.3 Intrinsic Evaluation Corpora

To conduct intrinsic evaluations of the embedding models, four distinct sub-datasets were utilized to assess four different aspects: semantic word similarity (S_s), syntactic word similarity (S_y), relatedness (R_r), and analogy tasks (A_t). The semantic, syntactic, and relatedness similarity datasets are structured as triplets. For example, the semantic similarity dataset is represented as $S_s = (W_{s_i}, W_{s_{i+1}}, H_{s_i})$, where the first two entries denote semantically similar word pairs and the third entry represents the average human judgment score. Similarly, the syntactic similarity dataset triplet is represented as $S_y = (W_{y_i}, W_{y_{i+1}}, H_{y_i})$, indicating syntactically similar word pairs along with the human average similarity score. The relatedness dataset is defined as $R_s = (W_{r_i}, W_{r_{i+1}}, H_{r_i})$, where the first two entries represent word pairs related in meaning, and the third entry denotes the average human judgment score. The subscript i corresponds to the i^{th} word pair, with $i = 1, 2, 3, ..., Z$. The total number of triplets in S_s, S_y, and R_r is denoted as Z, and for embedding model generation, $Z = 100$ triplets have been constructed.

Regarding the analogy tasks, the dataset A_t is represented as $A_t = W_{ai} : W^*_{ai} :: W_{bi} : W^*_{bi}$, where the first word pair is semantically or syntactically related, and the second-word pair is associated with the first. The subscript i designates

the i^{th} analogy in A_t. The extrinsic evaluation corpus comprises 60,000 text documents out of a total of 100,000 text documents, with the remaining 40,000 text documents earmarked for intrinsic evaluations.

In the initial phase, 146 words were selected from the 40,000 text documents for both semantic ($W_{s_1} - W_{s_{146}}$) and syntactic ($W_{y_1} - W_{y_{146}}$) categories. Two annotators independently selected another set of 146 words for each category, providing similarity scores. However, these words were not included in the scope of the intrinsic evaluation dataset (i.e., the 40,000 documents). If the absolute score difference between the first and second annotators was less than 3, the word pairs were included in the corresponding datasets. Based on this technique and Cohen's kappa score [111], we selected 100-word pairs for both semantic and syntactic categories. A similar approach was followed for relatedness similarity datasets, resulting in the selection of 100-word pairs out of 123. The analogy tasks datasets were selected based on agreement between annotators and their kappa scores. In total, 100 analogy tasks were selected from 150 analogies. The intrinsic datasets and their corresponding kappa scores are provided in Table 3.16.

Table 3.16: Kappa scores for the intrinsic datasets.

Datasets	Kappa score (%)
Semantic Similarity (S_s)	71.55
Syntactic Similarity (S_y)	65.30
Relatedness (S_r)	75.13
Analogy Tasks (A_t)	56.90

In this context, S_s, S_y, and R_s corpora demonstrate substantial agreement, while A_t exhibits moderate agreement.

3.4.4 Online Bengali Text Classification Corpora

Recently, several corpora have emerged in the field of Bengali text classification. However, these online corpora lack standardization. In this study, we address this issue by standardizing four open-access Bengali corpora, such as BARD [104], OSBC[1], IndicNLP [105], and ProthomAlo[2], to evaluate the performance of AFuNet. While the initial step, data collection, has already been completed, this research focuses on refining and standardizing the existing corpora. Each corpus is randomly divided into training and testing sets with a 70:30 ratio and subsequently subjected to purification through the following steps:

[1]https://www.kaggle.com/datasets/ratulcse10/open-source-autonomous-bengali-corpus
[2]https://www.kaggle.com/datasets/twintyone/prothomalo

- Elimination of non-Bengali digits and characters, replaced with NULL values.
- Removal of punctuation, represented by NULL values.
- Successive instances of newlines and white spaces are consolidated into a singular newline and white space.
- Substitution of hashtags, HTML tags, and URLs with a single space.
- Extraction of duplicate texts.
- Substitution of single-character words with NULL values.

Following this purification and standardization process, specific text files become empty, while others contain only a single word. These types of text files are subsequently excluded from the training and testing sets. Consequently, specific text samples are omitted from the standardized corpus after purification. Table 3.17 provides a comprehensive summary of these five standardized corpora.

The statistical summary in Table 3.17 highlights significant aspects of the four Bengali text classification corpora: BARD, OSBC, IndicNLP, and ProthomAlo. Among these corpora, BARD stands out with 263,284 training samples and 112,921 testing samples, showcasing its substantial size. The BARD boasts the highest number of unique words in its training set (45,050,236 words), suggesting a wide vocabulary range. On the other hand, IndicNLP demonstrates a relatively compact training set with 2,753,423 words, making it a more focused resource. Regarding test sets, BARD exhibits the highest number of unique words (18,476,045 words), indicative of its diverse language usage. These distinctions underline the importance of considering training and test sample characteristics, including size and vocabulary, for effective text classification model development and evaluation.

Figure 3.9 illustrates the distribution of text samples across categories for the three corpora: BARD and IndicNLP. Figure 3.9a presents explicitly the distribution of text samples in the BARD corpus by category. Notably, the Sports category contributes the highest number of instances, accounting for 64.55% (242,852), while the Entertainment category comprises the smallest share, representing 5.05% (18,982) of the total samples.

Figure 3.9b represents the category-wise summary of the IndicNLP corpus. Notably, the Kolkata category contributes the highest number of samples, accounting for 40.83% (5,768), whereas the International category has the lowest

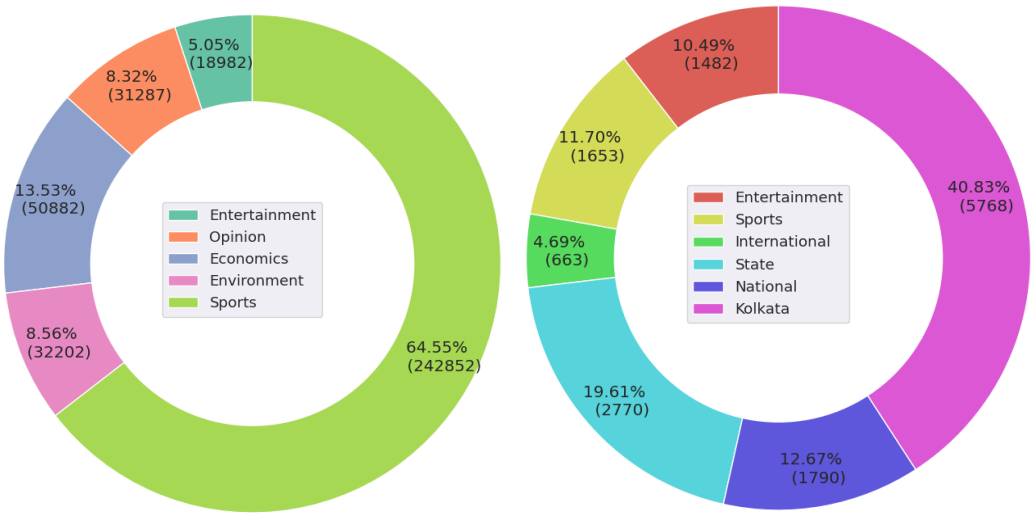
Table 3.17: Statistical summary of text classification corpora

Attribute	Value			
	BARD	OSBC	IndicNLP	Prothom-Alo
No. of samples	376205	95855	14126	213776
No. of category	5	12	6	8
No. of training samples	263284	66918	11300	149888
No. of testing samples	112921	28937	2826	63888
No. of words in train set	63968052	20650271	2753423	46911812
No. of unique words in train set	43050236	13225334	1840387	30407097
No. of words in test set	27484825	8869994	689804	20062786
No. of unique words in test set	18476045	5693743	459493	12984164
Max. words in train set sample	9658	11113	2466	9702
Min. words in train set sample	3	2	20	3
Max. words in test set sample	8435	11752	1936	9716
Min. words in test set sample	5	2	22	2
Avg. number of words	182.56	193.61	186.41	204.96

contribution with only 4.69% (663) of the total samples. It's worth mentioning that the IndicNLP corpus is the smallest among the five corpora in terms of the number of text samples. Figure 3.10a visualizes the ProthomAlo corpus distribution of text samples across categories. Among the 213,776 text samples, the Sports category accounts for the highest contribution with 28.00% of 59,847 samples, while the Education category has the lowest representation with 5.61% of 11,997 samples.

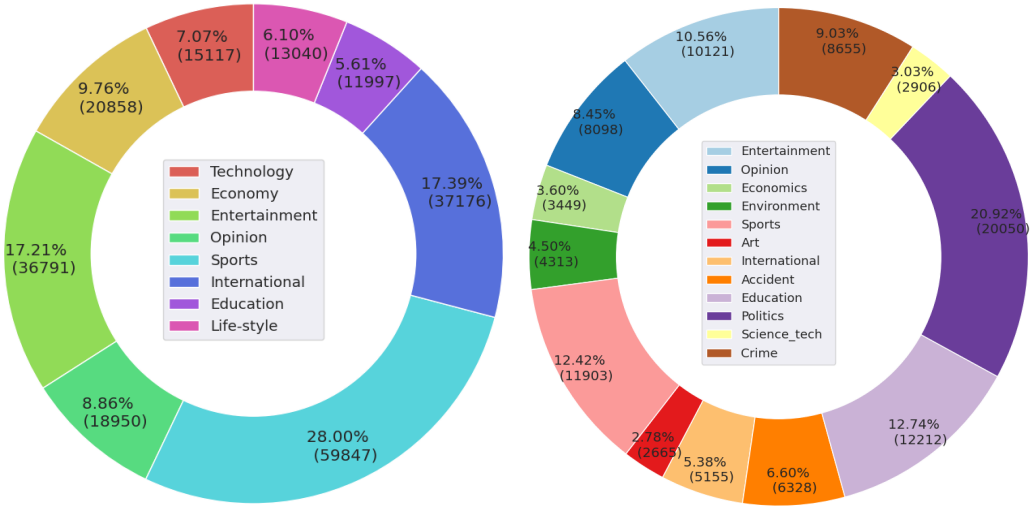
Figure 3.10 provides insights into the category-wise distribution of text samples within the Prothom-Alo and OSBC corpora.

Figure 3.10b depicts the category-wise distribution of the OSBC corpus. The OSBC corpus encompasses 12 distinct categories, comprising 95,855 text samples. Among these categories, the Politics category holds the highest number of samples with 20.92% (20,050), while the Science_tech category has the lowest representation at 3.03% (2,906) of the total samples. The OSBC corpus is



(a) BARD Distribution **(b)** IndicNLP Distribution

Figure 3.9: BARD, and IndicNLP corpora with category-wise text samples distribution



(a) Prothom-Alo Distribution **(b)** OSBC Distribution

Figure 3.10: BTCC1 and OSBC corpora with class-wise text samples distribution

notably characterized by a high degree of class imbalance across its categories.

3.4.5 Online Bengali Authorship Attribution Corpora

To evaluate the AuthorNet performance, this study used three online Bengali authorship attribution, i.e., Litratry Dataset (LD) [128], Bengali Authorship Attribution Dataset for 16 authors (BAAD16) [36], and Bengali Authorship Attribution Dataset for 6 authors (BAAD6) [36].

LD: The LD is the Bengali literature corpus. It contains three prominent Bengali authors, e.g., *Bankim Chandra Chattopadhyay* (1), *Rabindranath Tagore* (2) and *Sarat Chandra Chattopadhyay* (3) [128]. A total of 3000 literature texts split into 2,250 training texts and 750 texting texts, e.g., $C_1 =< \mathcal{T}_1, \mathcal{A}_1 >$ and $\mathcal{T}_1 = [1, ..., 3000]$, $\mathcal{A}_1 = [1, ..., 3]$). The corpus summary is shown in Table 3.18. Author (1) follows the Bengali Sadhu-bhasha linguistic variation and a few words from Sanskrit-bhasha. At the same time, authors (2) and (3) follow the Sadhu-bhasha and Cholito-bhasha linguistic variations.

Table 3.18: Statistics of Authorship Classification Corpora (LD, BAAD6, BAAD16)

Attributes	BAAD16	LD	BAAD6
#Author texts	17,966	3,000	2,100
#Author	16	3	6
Corpora size (MB)	210.8	21.6	37.1
#Words (M)	13.47	1.25	2.30
#Unique words (M)	0.59	0.08	0.17
#sentences (M)	0.71	0.08	-
#Tran author text	14,374	2,250	1,800
#Test author text	3,592	750	300
Max./Min. train author text size in KB	16.3/9.6	31.4/1.4	222.75/0.05
Max./Min. test author text size in KB	14.6/10.0	37.5/1.2	95.56/0.05

BAAD16: The BAAD16 is a Bengali multi-domain corpus (e.g., Novels, Stories, Series, and Blogs) and contains 16 different authors. The author's writing style follows the Bengali Cholito-bhasha and Sadhu-bhasha linguistic variations. Each of the texts contains 750 words and 14,374 training texts, and 3,592 inferencing texts, e.g., $C_3 =< \mathcal{T}_3, \mathcal{A}_3 >$ and $\mathcal{T}_3 = [1, ..., 17966]$, $\mathcal{A}_3 = [1, ..., 16]$. Table 3.18 shows the BAAD16 corpus summary. The author's names are encoded with an integer value, e.g., *Bankim Chandra Chattopadhyay (1)*, *Humayun Ahmed (2)*, *Manik Bandopadhyay (3)*, *Muhammed Zafar Iqbal (4)*, *Kazi Nazrul Islam (5)*, *Nihar Ranjan Gupta (6)*, *Rabindranath Tagore (7)*, *Shirshendu Mukhopadhyay (8)*, *Samaresh Majumdar (9)*, *Shordindu Shekhor Chakma (10)*, *Sarat Chandra Chattopadhyay (11)*, *Satyajit Ray (12)*, *Sunil Gangopadhyay (13)*, *Tarasankar Bandyopadhyay (14)*, *Taslima Nasrin (15)* and *Zahir Raihan(16)*.

BAAD6: The BAAD6 is the Bengali Blog-based corpus and contains six bloggers texts [129]. Most of the context of texts are religion, politics, sports, ed-

ucation, and lifestyle. There are 1800 training and 300 inferencing texts, e.g., $C_4 =< \mathcal{T}_4, \mathcal{A}_4 >$ and $\mathcal{T}_4 = [1, ..., 2100]$, $\mathcal{A}_4 = [1, ..., 6]$. Due to copyright and privacy issues, the corpus authors' actual names are not exposed. The anonymous author names are *FE (1)*, *HM (2)*, *EJ (3)*, *MT (4)*, *RN (5)* and *RG (6)*. Table 3.18 shows the summary of BAAD6. All bloggers follow the Bengali Cholito-bhasha linguistic variation, and a few words are taken from Sadhu-bhasha and English words with Bengali transliteration format.

3.5 Summary

Chapter 3 provided a comprehensive overview of the corpora development process. This chapter focused on creating two distinct types of corpora: embedding corpora, and text classification corpora. The text classification corpora were assembled using a manual annotation and verification approach, which proved time-consuming and costly. In response, we introduced an active learning-based text classification corpora development system to streamline manual annotation and reduce costs. In total, we developed five embedding corpora, seven text classification corpora, and one intrinsic evaluation corpora. We have also used four online Bengali text classification corpora performances by the proposed system (i.e., AFuNeT). The embedding corpora will be employed in the following chapter to develop embedding models, while the intrinsic evaluation corpora will serve as a valuable resource for evaluating these models.

Chapter 4

Embedding Model Development

An embedding model, often called a word embedding model, is a natural language processing (NLP) technique used to represent words or phrases as numerical vectors in a continuous vector space. These models are designed to capture the semantic and contextual information of words, making them suitable for various NLP tasks. In this chapter, we provide an in-depth exploration of two vital techniques: non-contextual and contextual embeddings. Furthermore, we furnish a comprehensive overview of the processes involved in crafting and evaluating these models, emphasizing their significance for downstream tasks.

4.1 Embedding Models

Developing low-resource embedding models involves tailoring word representations and language models for languages with limited resources. These models are crucial in improving natural language processing tasks for languages with less available data. Considering their semantic, syntactic, and context-aware feature extraction capabilities, the embedding model development technique can be categorized into two types: (i) Non-contextual embeddings technique and (ii) Contextual embeddings technique. Figure 4.1 represents the abstract view of embedding model development. In this study, we have crafted three distinct categories of non-contextual embedding models: Individual embedding models, Meta-Embedding models, and Attention-based Meta-Embedding models. In contrast, contextual language models were not built from the ground up; instead, we harnessed pre-trained language models for the purpose of text-to-feature extraction.

Figure 4.1, the initial stage of embedding model hyperparameter tuning takes as input the embedding corpora and the initial individual embeddings, yielding the top-performing embeddings as outputs. The selection of these top-performing

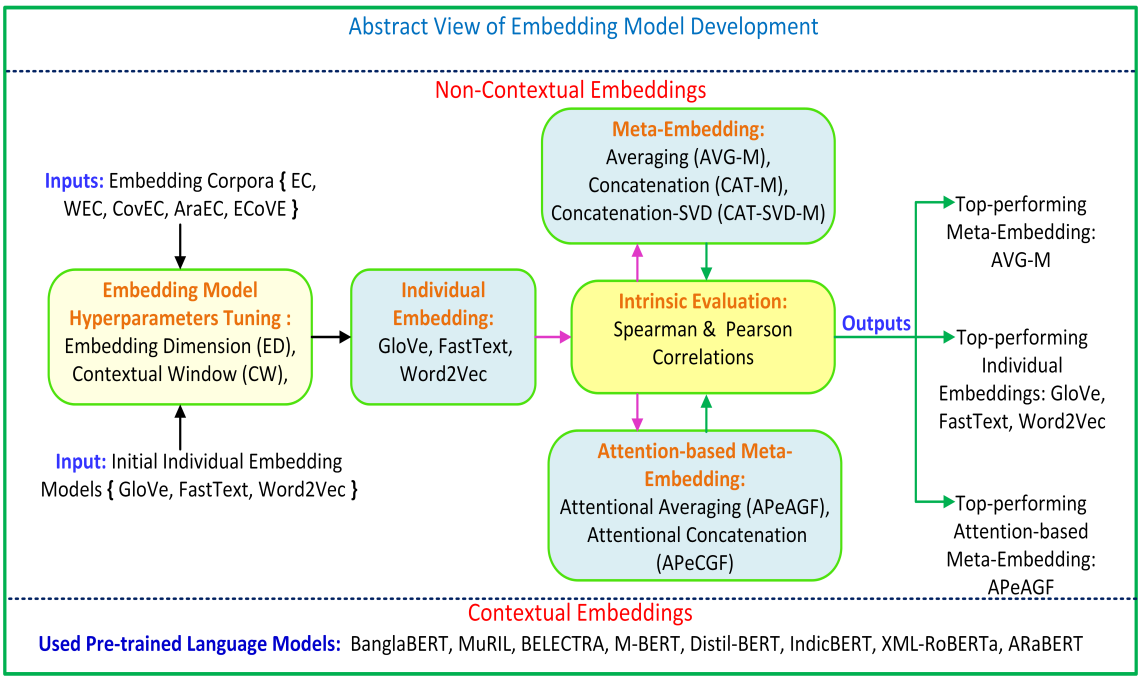


Figure 4.1: Abstract view of Embedding Model Development

embeddings is based on intrinsic evaluations. The following subsections will furnish a comprehensive breakdown of each of these methodologies.

4.2 Non-contextual Embeddings

Non-contextual embeddings, also known as static word embeddings, represent words or phrases based solely on their intrinsic characteristics and without considering the context in which they appear. These embeddings are pre-trained on large corpora and capture the underlying relationships between words, making them useful for various NLP tasks such as word similarity, sentiment analysis, and named entity recognition. Non-contextual embeddings, like Word2Vec, fastText, and GloVe, provide a fixed representation for each word, making them computationally efficient and applicable to low-resource languages. However, they may not capture nuances in word meaning that depend on the context, and their performance can be limited for tasks requiring context-aware understanding. In this study, we have devised a method for fine-tuning hyperparameters of embedding models using Bengali unlabeled corpora. The subsequent subsection elaborates on the specifics of this hyperparameter tuning approach.

4.2.1 Embedding Models Hyperparameters Tuning

In this study, we present an automatic hyperparameter tuning method tailored for low-resource Bengali text data, primarily focusing on intrinsic evaluation as the objective function. Several embedding techniques, such as GloVe, Word2Vec, FastText, and m-BERT, have yielded exceptional performance in English text language, and their effectiveness in downstream tasks has been well-documented. However, regarding low-resource languages like Bengali, the availability of optimized parameter sets for embedding models is limited. While certain parameters like embedding dimension (ED), contextual window (CW), word frequency, learning rate, epoch number, and context type are common across various embedding techniques, the key influencers of semantic and syntactic feature representation are ED, CW, and minimum word frequency (min_count). These parameters' interactions impact classification accuracy, and the multitude of potential combinations is overwhelming to explore manually.

To address this challenge, our study introduces the Embedding Parameters Identification (EPI) Algorithm (Algorithm 5) to identify an optimal set of parameters from the embedding corpus. Algorithm 4, a component of Algorithm 3, employs an objective function to maximize semantic and syntactic similarity accuracy within the given corpus. Specifically, our proposed EPI Algorithm 3 considers 89 ED values, 24 CW values, and 96 minimum word frequencies, resulting in a staggering 205,056 parameter combinations (ED: 89 x CW: 24 x min_count: 96). Out of these, Algorithm 3 efficiently selects the best 33 parameters (ED: 11, CW: 3, min_count: 1) for downstream tasks like text classification, significantly reducing the time required for model generation.

The process of generating embedding models commences with an embedding corpus denoted as EC, which comprises a collection of unlabeled text embeddings. This corpus can be represented as $EC : EC = et_1, et_2, et_3, ..., et_D$, where D represents the total number of embedding texts within EC. For the purposes of this research, the total number of embedding texts (D) amounts to 969,000. The proposed system employs three distinct word embedding techniques: GloVe, FastText, and Word2Vec. The subsequent subsections elaborate on the techniques employed in generating these embedding models.

GloVe The GloVe technique is employed to generate a total of 33 embedding models using the embedding corpus EC. This is achieved by varying the embedding dimension (ED) and context window (CW) with ED taking values from the set 25, 50, 100, 150, 200, 250, 275, 300, 325, 350, 400 and CW from 6, 9, 12. The remaining hyperparameters are kept at their default settings, which include

Algorithm 5: Automatic embedding parameters identification (EPI) based on intrinsic accuracy (e.g., semantic & syntactic)

```
Input : Embedding corpus (EC)
Output : Optimize hyperparameters
ED ← {}
CW ← {}
min_count ← 0
X_MAX ← 95
gSE ← 0, gSY ← 0
Function : EPI(EC) // Input Function {
SE ← 0, SY ← 0
for i ← 10 → 450 do
    SeAcc ← 0, SyAcc ← 0
    for j ← 2 → 25 do
        SeAcc1 ← 0, SyAcc1 ← 0
        for k ← min_count → X_MAX do
            model ← EmbeddingTechnique(EC, i, j, k)
            SticAc, SyticAc ← intrinsicEvaluation(model, SE, ST)
            SeAcc1, SyAcc1 ← max(SeAcc1, SyAcc1, SticAc, SyticAc)
        end
        SeAcc, SyAcc ← max(SeAcc1, SyAcc1, SeAcc, SyAcc, 1)
        ED, CW, min_count ← Update(SeAcc, SyAcc, i, j, k)
    end
    SE, SY ← max(SeAcc, SyAcc, SE, SY)
     ED, CW, min_count ← Update(SE, SY, i, j, k, 2)
end
ED, CW, min_count // Output: Optimal Hyperparameters }
```

min_count : 2, X_MAX : 95, *epochs* : 30, *binary* : *true*, *memory* : *true*, and *threads* : 8. GloVe's approach to generating feature vectors is based on global and local word counts, word-word co-occurrence, and local context with the central word. These characteristics enable GloVe to extract high-quality semantic and syntactic features, albeit at the cost of increased computation time due to matrix factorization.

FastText (Skip-gram & CBOW) FastText, a prediction-based embedding technique that leverages sub-word information, is utilized in this research. This technique takes the embedding corpus *EC* as its input and employs the gensim library for generating embedding models [130]. FastText, an extended version of Word2Vec, is capable of handling sub-word information. Two variants of FastText are employed: the skip-gram with negative sampling (with *negative* : 25) and the CBOW with hierarchical loss. A total of 66 embedding models (comprising 33

Algorithm 6: Update the embedding dimension, contextual window and min frequency words based on global semantic and syntactic accuracy

```
Function : Update(semacticAc, syntacticAc, i, j, k, Flag) // Input Function
{ if Flag == 1 then
    if fabs(gSE − semacticAc) < 1e − 3 then
        if fabs(gSY − syntacticAc) < 1e − 3 then
            gSE ← semanticAc, gSY ← syntacticAc
            CW.append(j)
        end
if Flag == 2 then
    if fabs(gSE − semacticAc) < 1e − 3 then
        if fabs(gSY − syntacticAc) < 1e − 3 then
            gSE ← semanticAc, gSY ← syntacticAc
            ED.append(i)
            min_count ← k
return ED, CW, min_count //Output: Optimized parameters sets }//
```

for skip-gram and 33 for CBOW) are generated by varying the hyperparameters, specifically the embedding dimension (ED) and context window (CW). The choices for ED include values from the set 25, 50, 100, 150, 200, 250, 275, 300, 325, 350, 400, and CW ranges over 6, 9, 12. The remaining hyperparameters are maintained at their default settings, such as min_count : 2, $epochs$: 30, a learning rate (lr) of 0.098, minimum n-gram 2, and maximum n-gram 6. The computational time required varies from a minimum of two hours for CW : 6 and ED : 25 to a maximum of thirty-one hours for CW : 12 and ED : 300.

Word2Vec (Skip-gram & CBOW): Word2Vec, a prediction-based embedding technique, generates embedding vectors from the center word to the context word or vice versa. In this research, Word2Vec processes the input embedding corpus EC to produce embedding models as output using the gensim library. A total of 66 embedding models are generated, with 33 each for the Skip-gram and CBOW variants. These models result from combinations of hyperparameters for the embedding dimension (ED) and context window (CW). Specifically, ED can take values from the set 25, 50, 100, 150, 200, 250, 275, 300, 325, 350, 400, and CW varies within 6, 9, 12. For the Skip-gram model, negative sampling ($negative$: 25) and 30 epochs are employed, while the CBOW model uses hierarchical loss. The remaining hyperparameters are set to their default values, including min_count, learning rate (lr), and threads.

Each embedding technique offers its own set of advantages and disadvantages.

GloVe, for instance, incorporates the entire corpus statistics during word feature extraction, in contrast to Word2Vec, which relies on local window-based statistics. GloVe considers both local and global word frequency as well as word-to-word co-occurrences, while FastText only takes local word frequency into account. GloVe typically requires less training time than m-BERT but consumes more memory during matrix factorization compared to Word2Vec. Additionally, a high frequency of stop words can impact the feature values in GloVe embedding but not in FastText embedding. GloVe focuses on the local context of a word, whereas m-BERT assesses the complete context of a word in the corpus. In total, the three embedding techniques can generate 165 embedding models, with 33 for GloVe, 66 for FastText, and 66 for Word2Vec. These models are subjected to intrinsic evaluation (as discussed in Section 6.4.1). Among them, the top-performing 3 embedding models are selected for document classification purposes (as described in Section 6.4). It is important to note that the default hyperparameters of GloVe, FastText, and Word2Vec algorithms may not be suitable for all types of text processing, highlighting the need for hyperparameter tuning prior to model training.

In the field of Bengali text classification, limited research has been conducted on developing non-contextual embedding models. However, the development and selection of optimal non-contextual embedding models are pivotal for enhancing the performance of low-resource text classification. To address this, we have created a system for embedding model development and intrinsic evaluation. Our research encompasses three categories of non-contextual embedding model development and intrinsic evaluation: (i) Individual embedding and intrinsic evaluation, (ii) Meta-embedding and intrinsic evaluation, and (iii) Attention with Positional Meta-embedding and intrinsic evaluation. Each of these categories is discussed in the subsequent subsections.

4.2.2 Individual Embedding and Intrinsic Evaluations

Individual embedding refers to the development of separate embedding models for a specific purpose, language, or dataset. Intrinsic evacuation helps determine how well these models represent and understand language patterns and semantics. In this study, three embedding techniques, namely Word2Vec, GloVe, and FastText, were employed. The hyperparameters of these embedding methods underwent tuning for five distinct embedding corpora: Bengali embedding corpus (BEC), Authorship embedding corpus (WEC), Bengali Covid embedding corpus (CovEC), English embedding corpus (ECoVE), and Arabic embedding corpus

(AraEC), utilizing the Embedding Parameters Identification (EPI) Algorithm outlined in Algorithm 5.

4.2.2.1 Individual Embedding Models Training

Using an unlabelled Bengali embedding corpus, BEC, we generated a collection of individual embedding models employing three different embedding techniques: Word2Vec (utilizing Skip-gram and CBOW) [7], FastText (employing Skip-gram and CBOW) [9], and GloVe [8]. The optimization of embedding hyperparameters for the Skip-gram and CBOW variants of FastText and Word2Vec is achieved through the Embedding Parameter Identification (EPI) method [2]. The Python-based Gensim library [130] was utilized for generating these embedding models. Default hyperparameters often prove ineffective in Bengali text classification tasks due to the language's high inflection rate and morphological variations, including variations like Sadhu-bhasha, Cholito-bhasha, and Sanskrit-bhasha. To address this issue, the EPI method was employed to select the most suitable hyperparameters [2]. Table 4.1 displays the initial and optimized hyperparameters of the

Table 4.1: Initial and optimized hyperparameters of GloVe, fastText and Word2Vec for BEC and WEC

Model	Initial hyperparameters	Optimized hyperparameters
Word2Vec, FastText and GloVe	ED: {100, 150, 200, 250, 300}, CW: {3, 6, 9, 12}, V: {Skip-gram, CBOW}, M-F: {1, 2, 3}, N-S: {15, 20, 25} X-Max: {70, 80, 95, 100} Epoch: {1-50}	ED: {200, 250, 300}, CW: {9, 12}, V: {Skip-gram}, M-F: {2}, N-S: {25}, X-Max: {95}, Epoch: {33}

three local contextual embedding models. The optimized embedding dimensions (ED) for these methods are {200, 250, 300}, and the contextual window (CW) sizes are {9, 12}. Among the three embedding methods, the Skip-gram version (V) demonstrates superior performance for FastText, Word2Vec, and GloVe, with a minimum word frequency set to {2}. Both FastText and Word2Vec employ a negative sample cut-point of 25, while GloVe's maximum allowable frequency is set to 95. Additionally, FastText and Word2Vec models utilize a negative sampling value of 25, and the three embedding methods converge after 33 epochs for the unlabelled corpus, BEC. Similar optimization processes were applied to generate other embedding models. Table 4.2 provides a summary of the optimized parameters for the CovEC, ECoVE, and AraEC embedding corpora.

Minimum n-grams (Mgs) are applied during the FastText training phase, which splits each word based on this value. FastText and Word2Vec create em-

Table 4.2: Optimized hyperparameters for CovEC, ECoVE, and AraEC

Methods	Optimized Hyperparameters
Word2Vec (SG), FastText (SG) and GloVe	ED: {200, 250, 300}, Min.Frequency: 2, Window Size: {12, 13}, Max.Frequency: 75, Epoch: 25, Mgs: 2, learning_rate: 0.01

bedding models using a "center word to context word prediction" approach, while GloVe employs "word-word co-occurrence and frequency" techniques. In this study, we explore three different embedding dimensions (200, 250, 300), two context window sizes (12 & 13), and three embedding methods (GloVe, FastText, and Word2Vec), resulting in a total of 18 unique embedding models generated for intrinsic evaluation. GloVe, a statistical word frequency-based method, and neural embedding-based methods like Word2Vec and FastText are trained using optimized hyperparameters (refer to Table 4.2). The Skip-gram version of FastText excels in handling sub-word information during the embedding model training, which proves particularly beneficial for morphologically rich languages, mitigating out-of-vocabulary (OOV) issues. We generate a total of 18 individual embedding models (6 for Word2Vec, 6 for FastText, and 6 for GloVe), representing various combinations of 3 embedding dimensions (ED) and 2 window sizes. All these models are employed for intrinsic evaluation purposes.

Intrinsic Evaluation of Individual Embedding In this step, individual embedding models are taken as inputs, and the outcome consists of the best-performing models. Three embedding models are generated, with the selection of one best-performing model from each method (GloVe, FastText, and Word2Vec). These top-performing models are chosen through intrinsic evaluation, which assesses the quality of an embedding model for specific NLP tasks. Intrinsic evaluators not only help gauge model quality but also contribute to reducing downstream task training time and mitigating out-of-vocabulary (OOV) issues, as described in Algorithm 7.

In Algorithm 7, the function $HumanJudgementScore(.)$ provides the pairwise annotator scores for the semantic (H_m), syntactic (H_y), relatedness (H_r), and analogy tasks (H_{at}) datasets. Each embedding model (em) is assessed using four datasets ($S_{s\hat{\rho}}$, $S_{y\hat{\rho}}$, $S_{r\hat{\rho}}$, and $S_{at\hat{\rho}}$), where cosine similarity scores are computed for word pairs within each dataset. The Spearman correlation ($SprCor(.)$) and Pearson correlation ($PerCor(.)$) functions then utilize annotator judgment scores and cosine similarity scores to produce Spearman correlation ($\hat{\rho}$) and Pearson correlation ($\hat{r}$) values. Specifically, we have $S_{s\hat{\rho}}$, $S_{y\hat{\rho}}$, $S_{r\hat{\rho}}$, and $S_{at\hat{\rho}}$) for the

Algorithm 7: Intrinsic Evaluation for Word Embedding Models

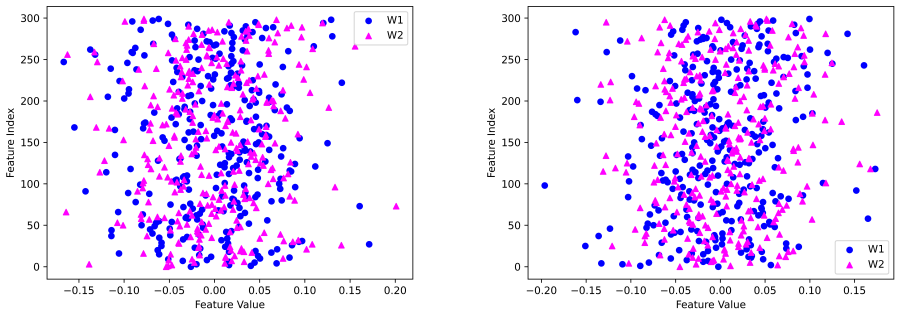
$Function : \Delta 3(S)$ // Intrinsic Evaluation Function {
$S_s := X_P$ *Semantic words pairs*
$S_y := X_P$ *Syntactic words pairs*
$R_s := X_P$ *Relatedness words pairs*
$A_t := X_a$ *Analogy tasks*
$H_m, H_y, H_r, H_{at} := HumanJudgementScore(S_s, S_y, R_s, A_t)$
$I_r = []$
for *ab in* $range(1, len(S))$ **do**
    $C_m := [],\ C_y := [],\ C_r := [],\ C_{at} = []$
    $em = S[ab]$
    for *i in* $range(1, X_P - 1)$ **do**
        $C_m.append(CosineSimilarity(S_s[i], S_s[i+1], em))$
        $C_y.append(CosineSimilarity(S_y[i], S_y[i+1], em))$
        $C_r.append(CosineSimilarity(R_s[i], R_r[i+1], em))$
    end
    for *i in* $range(1, X_a)$ **do**
        $temp = a_t[i]$
        $C_{at}.append(CosineSimilarity(temp[3], temp[4], em))$
    end
    $\{S_{s\hat{\rho}}, S_{y\hat{\rho}}, S_{r\hat{\rho}}, S_{at\hat{\rho}}\} := SprCor(H_m, C_m, H_y, C_y, H_r, C_r, H_{at}, C_{at})$
    $\{S_{s\hat{r}}, S_{y\hat{r}}, S_{r\hat{r}}, S_{at\hat{r}}\} := PerCor(H_m, C_m, H_y, C_y, H_r, C_r, H_{at}, C_{at})$
    $l_b := Pavg(S_{s\hat{\rho}}, S_{y\hat{\rho}}, S_{r\hat{\rho}}, S_{at\hat{\rho}}, \delta_m, S_{s\hat{r}}, S_{y\hat{r}}, S_{r\hat{r}}, S_{at\hat{r}})$
    $I_r.append(l_b)$
end
$B_a := best(I_r), a = [GloVe, FastText, Word2Vec]$
return B_a // Output }

semantic, syntactic, relatedness, and analogy tasks, while $S_{s\hat{r}}$, $S_{y\hat{r}}$,$S_{r\hat{r}}$, and $S_{at\hat{r}}$ represent the Pearson correlation scores. The $Pavg(.)$ function combines these six scores to compute the average score for the b^{th} embedding model's hyperparameters. In this manner, intrinsic evaluators assess all the embedding models, and the $best(.)$ function is employed to select the top-performing embedding models. Finally, the $\Delta 3(.)$ function yields the three best-performing embedding models (B_a). *The comprehensive evaluation of the intrinsic performance of individual embedding will be detailed in Section 6.4.1.*

4.2.3 Meta-embedding and Intrinsic Evaluations

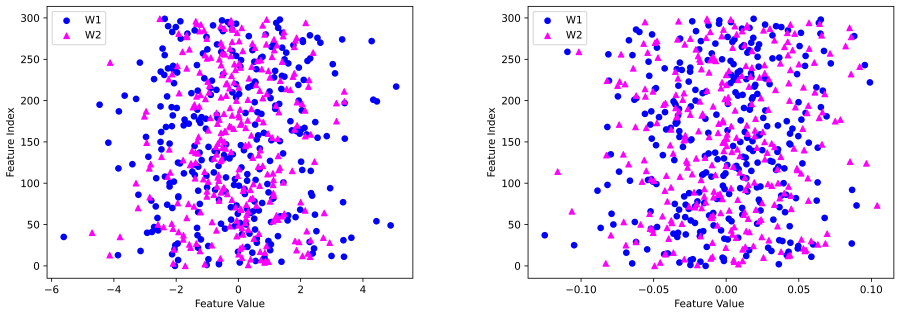
Meta-feature fusion integrates features from diverse individual embedding models to create a comprehensive combined representation. In this study, three Meta-Embedding (ME) methods, namely Concatenation with Singular Value Decomposition Meta-embedding (CONC-SVD-M), Concatenation Meta-embedding (CONC-M) Average Meta-embedding (AVG-M), are employed. These methods address the issue of different feature distributions produced by distinct embedding techniques, resulting in varying semantic and syntactic interpretations of the same word within a given corpus. The module takes the three best-performing individual embedding models, one from each embedding method, and generates three ME models. This variation in word meaning across different embedding models can have a significant impact on text classification tasks, as depicted in Fig. 4.2, which illustrates the feature distribution of two syntactically similar words, $W1$, and $W2$, using both individual embedding models and ME.

The AVG-M ME outperformed the individual Word2Vec embedding with a maximum syntactic cosine similarity of 96.83%, while the minimum cosine similarity of 92.04% was observed in the individual Word2Vec embedding. The standard deviations for two words, σ_{W1} and σ_{W2}, were minimal at 0.1707 and 0.1580, respectively, in the AVG-M feature distribution. In contrast, the Word2Vec feature distribution exhibited higher standard deviations, ranging from 0.1857 to 0.1868. Studies such as Madhyastha et al. [131] have confirmed that higher standard deviation values in embedding models can lead to decreased text classification accuracy. Our intrinsic evaluation results indicate that embedding models with stronger semantic and syntactic correlations results in improved text classification accuracy. Figure 4.2 visually displays the standard deviation and cosine similarity values in the proposed ME feature distribution, revealing that ME captures more coherent semantic and syntactic features, ultimately leading to higher accuracy in text classification tasks. Figure 4.3 illustrates the feature



(a) FastText feature distribution with σ_{W1}=0.1732, σ_{W2}=0.1738, $CS_{(W1,W2)}$=96.66%

(b) GloVe feature distribution with σ_{W1}=0.1801, σ_{W2}=0.1855, $CS_{(W1,W2)}$=95.16%



(c) Word2Vec feature distribution with σ_{W1}=0.1857, σ_{W2}=0.1868, $CS_{(W1,W2)}$=92.04%

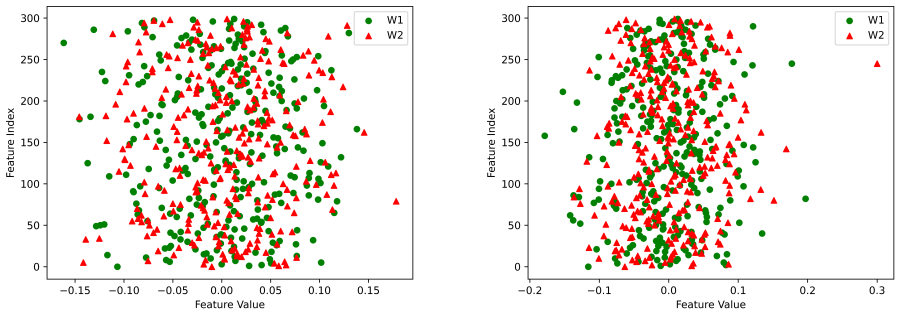
(d) ME (AVG-M) feature distribution with σ_{W1}=0.1707, σ_{W2}=0.1580, $CS_{(W1,W2)}$=96.83%

Figure 4.2: Syntactic word pair (W1 = Transliteration: Parbata, Bengali to English Translation: Mountain & W2 = Transliteration: Parbatamālā, Bengali to English Translation: Mountains) feature distribution. AVG-M used the FastText & GloVe

distributions of FastText, GloVe, Word2Vec, and AVG-M, showcasing their respective cosine similarity and standard deviation values for two semantic words (W1 and W2). The ME, specifically AVG-M, achieved the highest semantic word similarity score at 92.97% and the lowest semantic similarity at 89.07%.

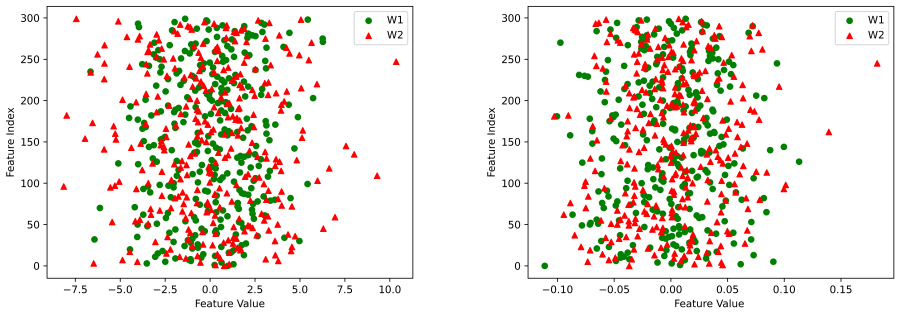
The AVG-M feature distribution yielded the minimum standard deviation values of 0.1535 for W1 and 0.1322 for W2. In contrast, the Word2Vec feature distribution exhibited higher maximum standard deviations of 0.1924 for W1 and 0.1780 for W2. These results demonstrate that AVG-M provides superior Bengali feature representations when compared to individual feature representations. For a more detailed understanding of the ME feature generation and evaluation techniques, please refer to Algorithm 8.

The Meta-Feature Generation and Evaluation (MFGE) Algorithm takes an embedding model (E) as input and generates a Meta-Embedding (ME) matrix



(a) FastText feature distribution with σ_{W1}=0.1814, σ_{W2}=0.1687, $CS_{(W1,W2)}$=91.77%

(b) GloVe feature distribution with σ_{W1}=0.1780, σ_{W2}=0.1490, $CS_{(W1,W2)}$=92.55%



(c) Word2Vec feature distribution with σ_{W1}=0.1924, σ_{W2}=0.1780, $CS_{(W1,W2)}$=89.07%

(d) ME (AVG-M) feature distribution with σ_{W1}=0.1535, σ_{W2}=0.1322, $CS_{(W1,W2)}$=92.97%

Figure 4.3: Semantic word pair (W1 = Transliteration: Abakasa, Bengali to English Translation: Vacation & W2 = Transliteration: Chuti, Bengali to English Translation: Holiday) feature distribution. AVG-M used the FastText & GloVe

with three distinct ME models (ME[1:3]). A total of nine ME models are developed through Algorithm 8, comprising three for AVG-M, three for CONC-M, and three for CONC-SVD-M. From these nine models, the three best-performing ME models are selected for downstream tasks such as text classification, as determined by intrinsic evaluation. *The comprehensive evaluation of the Meta-Embedding performance will be detailed in Section* 6.4.2.

4.2.4 Attention with Position-based Meta-embedding and Intrinsic Evaluation

Attention with Position-based Meta-embedding is a technique in natural language processing that combines meta-embedding methods with the concept of attention mechanisms. It enhances the representation of words in a text by considering both their semantic content and their positional information in the document.

Algorithm 8: Meta-Embedding feature generation and evaluation (MFGE)

$Function : MFGE(E)$ //Input: Best 3 Individual embeddings {
$L_e := len(E)$, $idx := 1$ //Measured the embedding numbers
for $i \leftarrow 1 \rightarrow L_e$ **do**
  for $j \leftarrow i+1 \rightarrow L_e$ **do**
    $ME[idx] := E[i : ED] \oplus E[j : ED]$
    $ME[idx = idx + 1] := \frac{\sum_{k=1}^{ED}(E[i:k], E[j:k])}{2}$
    $S_v := SVD(E[i : ED] \oplus E[j : ED])$
    $ME[idx = idx + 1] := S_v[1 : ED]$
  end
end
$ME := I_{ME}(ME)$
$return\ (ME[1 : 3])$ // Output }

This approach aims to capture the contextual relationships between words more effectively, improving performance in various NLP tasks, such as text classification, document retrieval, and machine translation. Attention to Position-based Meta-embedding has shown promising results in enhancing the understanding of textual data in low-resource languages, including Bengali and morphologically rich languages.

The best-performed individual embedding models $B_a \in \{GloVe, FastText, Word2Vec\}$ are used as the inputs and generate the fused feature matrix $(FM_q/FM_{q'})$, where $q \in (p^{lb} + q^{lb})$, $q^{'} \in (p^{ul} + q^{ul})$. Initially, the training sample (T^n) sequentially extracts the feature using the mapping function $M(.)$ and a^{th} embedding model. The a^{th} embedding model feature matrix for q^{th} training sample is represented by $FM_{qa} \in \mathbb{R}^{sl \times ED}$, where $sl \in 256$ denotes the maximum sequence length and $ED \in 300$ indicates the optimal embedding dimension.

The q^{th} training sample sequentially produces three feature matrices, e.g., FM_{q1} for $GloVe$, FM_{q2} for $FastText$ and FM_{q3} for $Word2Vec$. Whereas $q^{'th}$ testing sample produces three feature matrices denoted by $FM_{q'1}$ for $GloVe$, $FM_{q'2}$ for $FastText$ and $FM_{q'3}$ for $Word2Vec$. The word position is crucial information for the context-aware word-level semantic and syntactic feature representation [132]. The position-based information is added before applying the self-attention operation. The $q/q^{'}$ sample sinusoidal position encoding operation is conducted using Eq.4.1.

$$PE_{q/q'}[1 : sl] = \begin{cases} sin(\frac{pos}{10000^{(2\times pos/ED)}}),\ if\ (pos\%2)==0\ pos=1,...,sl \\ \\ cos\frac{pos}{10000^{(2\times pos+1/ED)}},\ otherwise\ pos=1,...,sl \end{cases} \tag{4.1}$$

Here, ED denotes the embedding dimension, and the word position of $q/q^{'}$ sample is $pos \in sl$. This position encoding is just added to the training/testing sample ($q/q^{'}$) and fed to the trainable self-attention block, i.e., $Attention(.)$. The self-attention block contains nine trainable weight matrices, e.g., three for GloVe, three for FastText, and three for Word2Vec. The generalized form of matrices are query ($W^{aQ} \in \mathbb{R}^{sl \times ED}$), keys ($W^{aK} \in \mathbb{R}^{sl \times ED}$) and values ($W^{FMqa} \in \mathbb{R}^{sl \times ED}$) for GloVe, FastText and Word2Vec respectively. However, the attention of q^{th} sample is calculated using Eq. 4.2.

$$FM_{qa}^{'} = (\frac{(FM_{qa} \times W^{aQ}) \times (FM_{qa} \times W^{aK})^{T}}{\sqrt{ED}}) \times (FM_{qa} \times W^{FMqa}) \tag{4.2}$$

Here, FM_{qa} indicates the input fused feature matrix whereas $FM_{qa}^{'}$ represent the attention-based output fused feature matrix of a^{th} best performing embedding model (i.e., $a = [1 : GloVe, 2 : FastText, 3 : Word2Vec]$). The addition and layer normalization block is combined with an attention-based and positional encoding feature, which improves the word-level correlation [132]. The layer-based normalized feature is forwarded to the feature fusion block and fuses the feature value. In the training phase, the fused feature is denoted by FM_q and used in the CNN training module, whereas the testing time fused feature is represented by $FM_{q'}$ and will be used for the CNN model evaluation purpose. *The comprehensive evaluation of Attention-based Meta-Embedding performance will be detailed in Section 6.4.3.*

4.3 Contextual Embeddings

Contextual embeddings are a type of word or phrase representation in natural language processing that captures the context in which words appear in text. Contextual embeddings are typically created using deep learning models known as transformers, such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer). These models are pre-trained on large text corpora and learn to predict a word based on the words that come before and after it. As a result, they generate word representations that consider the meaning and usage of words in their specific contexts.

In this research, we have used two types of pre-trained transformer-based language models such as (i) Monolingual transformer-based language models and (ii) Multilingual transformer-based language models. Both types of per-trained models are used for fine-tuning purposes.

4.3.1 Monolingual transformer-based language models

Monolingual models in this study refer to language models exclusively trained on a single language. The research utilizes three pre-trained monolingual transformer-based language models tailored for specific languages: Bangli-BERT for Bengali, Bengali-ELECTRA for Bengali (based on ELECTRA), and AraBERT for Arabic. These models are fine-tuned using Bengali text classification corpora and Arabic Covid text identification corpus. The fine-tuning process involves adapting hyperparameters to match the characteristics of the specific tasks and corpora, ultimately achieving state-of-the-art (SOTA) performance compared to default hyperparameters.

4.3.2 Multilingual transformer-based language models

Multilingual transformer-based language models, in this context, pertain to models trained using multiple languages. In this research, eight pre-trained multilingual transformer-based language models are employed for fine-tuning. These models include M-BERT, Bengali-BERT, M-Distil-BERT, Bengali-ELECTRA, Indic-BERT, MuRIL, XML-Roberta, and AraBERT. During the fine-tuning process, these pre-trained models are adjusted with hyperparameters tailored to Bengali text classification corpora and Arabic Covid text identification corpus.

4.4 Summary

Chapter 4 overviews embedding models, their fundamental principles, and their applications in resource-constrained text classification. This research explores two main techniques for embedding model development: non-contextual embedding and contextual embedding. Within the non-contextual embedding category, three models are introduced: Individual embedding, Meta-Embedding, and Attention-based Meta-Embedding. Each model within this category undergoes intrinsic evaluation. While individual embeddings fail to address out-of-vocabulary (OOV) word issues, Meta-Embedding and Attention-based Meta-Embedding effectively mitigate these problems through n-gram tokenization of FastText embeddings. However, it's essential to note that non-contextual embedding models are static and do not adapt word meanings based on contextual semantics. To address this limitation, this study introduces transformer-based language models, such as BERT and ELECTRA variations, tailored for resource-constrained languages like Bengali. These pre-trained transformer-based language models serve as the foundation for Bengali text-to-feature extraction and are fine-tuned using the

newly developed Bengali text classification corpora. Chapter 5 will demonstrate how these embedding models are leveraged for downstream tasks, specifically resource-constrained text classification model development.

Chapter 5

Development of Text Classification Models

Text classification models are machine learning, deep learning, or transformer-based models designed to automatically classify or categorize text documents into predefined classes or categories based on their content. Resource-constrained text classification model development is a critical endeavor in the field of natural language processing, particularly for languages facing limitations in resources and data. This research explores and leverages three fundamental techniques to address the challenges posed by resource constraints. These models are trained to understand the patterns and features in the text and make predictions about the category to which a given text document belongs. In this chapter, we will explore text classification techniques and provide insights into developing effective resource-constrained text classification models. We will also discuss the process of selecting the best-performing text classification model from a set of models. In the upcoming sections, various types of text classification models will be developed as part of this research. The models include:

- Bengali Authorship Attribution: Attribution models will be developed using the self-build BACC-18 corpus and three additional online available corpora: BAAD16, BAAD6, and LD.
- Bengali Text Classification: Classification models will be created using three self-built corpora, namely BDTCC, BTCC11, BCoVC, and four online available corpora: BARD, OSBC, IndicNLP, and Prothom-Alo.
- Arabic Text Classification: A classification model will be developed using the self-built Arabic COVID-19 text classification corpus (AraCoV).
- English Text Classification: The classification model for English text will

be developed using the self-built corpus, i.e., English COVID-19 text classification corpus (ECoVC).

5.1 Text Classification Techniques

Developing text classification models for resource-constrained Bengali language involves three distinct techniques: (i) Statistical Models, which rely on traditional statistical techniques like TF-IDF feature extraction and classical machine learning algorithms such as Naïve Bayes, Support Vector Machines, and Random Forests. (ii) Deep Learning Models, which harness neural networks with multiple hidden layers, including Convolutional Neural Networks (CNN) and recurrent Neural Networks (RNN), such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), to capture complex text patterns. (iii) Transformer-based models, leveraging state-of-the-art models in Natural Language Processing (NLP), such as BERT, ELECTRA, and their variants, fine-tuned for Bengali text classification. Figure 5.1 depicted the abstract view of text classification model development.

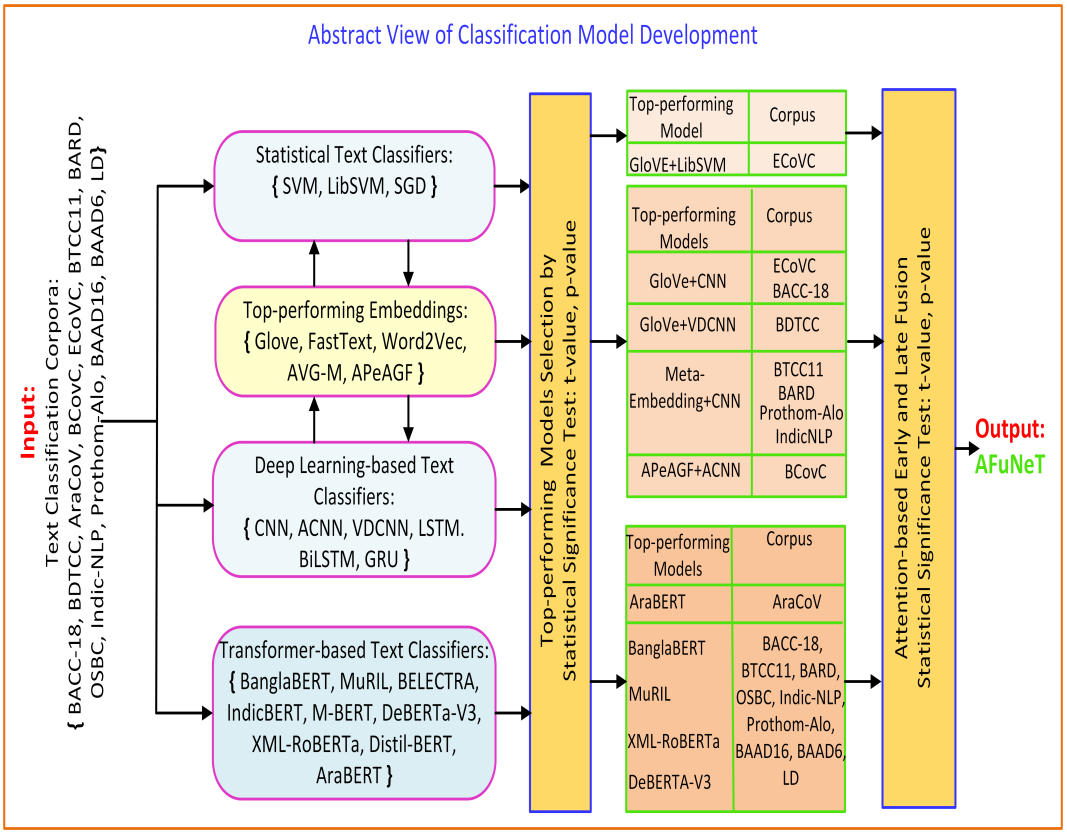


Figure 5.1: Abstract view of Text Classification Model Development: AFuNeT

In Figure 5.1, the inputs consist of the text classification corpora, and the

output is the best-performing text classification model (i.e., AFuNeT). The top-performing embeddings are employed in the development of both deep learning and statistical learning text classification models. Meanwhile, the transformer-based language models are fine-tuned using the pre-trained models. Once the classification models are developed, the deep learning classification models undergo evaluation using the corpora, and the statistical significance test is applied to select the top-performing text classification models. Similarly, transformer-based fine-tuned models and statistical models are chosen through the statistical significance test. Finally, the selected models are used for late fusion, resulting in the selection of the top-performing four models. These model combinations are employed in an attention-based early fusion framework. Ultimately, among these model combinations, the best-performing classification model is statically selected for resource-constrained text classification.

5.2 Statistical Text Classifiers

In the development of statistical classification models, linguistic features from each text document are first extracted using techniques such as term frequency (TF) and term frequency-inverse document frequency (TF-IDF). These methods are employed to transform text documents into numerical representations suitable for machine learning algorithms. The resulting feature values are then utilized in conjunction with traditional statistical classification methods, including SVM and LibSVM.

Support Vector Machine (SVM) & LIBSVM: SVM is a traditional and versatile supervised learning algorithm used for classification. Its primary objective is to find an optimal hyperplane that best separates two or more classes in a high-dimensional feature space. The key idea behind SVM is to identify support vectors, which are data points closest to the hyperplane. SVM then maximizes the margin between these support vectors, which results in better generalization to unseen data. In this research, we used the SVM for Bengali text classification corpora with the following parameters: kernel = sigmoid, tol = 0.00001, decision function shape= over, and max iteration=5.

LIBSVM is a specialized library designed for advanced support vector machine (SVM) implementation, offering efficient tools for SVM classification, regression, and distribution estimation. This library supports both linear and non-linear SVMs by utilizing a variety of kernel functions. LIBSVM is accessible to a wide audience of machine learning practitioners as it provides a Python-based imple-

mentation. In the Bengali text classification research, specific hyperparameters were selected to facilitate the development of classification models, including the use of a sigmoid kernel, a tolerance value of 0.00001, decision function shaping set to 'over,' and class weight adjustment with a 'balanced' parameter.

5.3 Deep Learning-based Text Classifiers

In this study, we explored Bengali text classification using six classification models including CNN, ACNN, VDCNN, LSTM, BiLSTM, and GRU as well as top-performing five embeddings, i.e., GloVe, FastText, Word2Vec, AVG-M, and APeAGF. Given the linguistic diversity of the Bengali language, hyperparameters from high-resource deep learning models cannot be directly applied to low-resource languages like Bengali. Consequently, our research initially focused on fine-tuning the hyperparameters of these deep-learning models. Subsequently, we developed classification models using the optimized hyperparameters. This research has chosen four classification models based on text classification accuracy (as discussed in Section 6.5). These models include (i) GloVe with CNN Model, (ii) GloVe with VDCNN Model, (iii) Meta-Embedding with CNN Model, and (iv) Attention-based Meta-Embedding with ACNN Model. The next subsection outlines our approach to hyperparameter tuning for deep learning-based models and provides insights into these four top-performing models.

5.3.1 Hyperparameters Tuning

In this study, we explored the performance of hyperparameter tuning for various models, including CNN, LSTM, BiLSTM, and GRU, in the context of Bengali text classification. Among these models, the CNN-based approach demonstrated the best performance. The CNN-based text classification framework, as described by Kim [12], was employed to train the feature matrix (MF_{ij}^{l}) and generate the CNN classification model. It's important to note that hyperparameters for classification models need to be tuned specifically for the Bengali text language being used [2]. To address this, we developed an Automatic Hyperparameters Tuning and Selection (AHTS) algorithm tailored for CNN and conducted experiments on Bengali text classification tasks. Insights of the Algorithm is presented in Algorithm 9.

The AHTS algorithm commences by selecting the optimal batch size (B_S) and maximum word sequence length (S_L), as these two hyperparameters significantly influence the CNN model and its tuning phases [133]. In the next stage, it

Algorithm 9: Automatic hyperparameters tuning and selection (AHTS) for CNN framework

Function : $AHTS(C_f)$ // Input Function {
  $\Delta_A := 0.0, B_S := 0, S_L := 0, M_B := 256, M_{sl} := 512$
for $i \leftarrow 5 \rightarrow M_B, i+ = 5$ **do**
  for $j \leftarrow 50 \rightarrow M_{sl}, j+ = 5$ **do**
    $ac := CNN(i, j, C_f)$
    if $ac > \Delta_A$ **then**
      $\Delta_A := ac, B_S := i, S_L := j$
  end
end
$F_C := [[2, 3, 4], [2, 3, 4, 5], [3, 4, 5], [3, 4, 5, 6], [2, 3, 4, 5, 6]]$
$F_S := [], \Psi = [], K_S := 0, L_r := 0.0, M_{ks} = 256, M_{lr} := 1.0, \Delta_A := 0$
for i *in* F_C **do**
  for $j \leftarrow 1 \rightarrow M_{ks}$ **do**
    for $k \leftarrow 0.1 \rightarrow M_{lr}, k+ = 0.1$ **do**
      $ac := CNN(B_S, S_L, i, j, k, C_f)$
      if $ac > \Delta_A$ **then**
        $\Psi[1] := i, \Psi[2] := j, \Psi[3] := k, \Psi[4] := B_S \Delta_A := ac$
    end
  end
end
return (Ψ) //Output }

proceeds to determine the number of filters (K_S), individual 1D filter sizes (F_S), and learning rate (L_r). Subsequently, the proposed AHTS algorithm fine-tunes the best five hyperparameters ($\Psi = B_S$, S_L, F_S, K_S, L_r) using BDTC and then applies the same hyperparameters to the other corpora (i.e., BACC-18, BTCC11 and BCovC). For experimental purposes, the CNN hyperparameters are also tuned using Keras Tuner [134].

5.3.2 GloVe with CNN-based Model: Bengali Authorship Attribution

The proposed authorship attribution framework consists of four key modules: embedding model generation, feature representation, classifier training, and classifier testing. The embedding model generation function constructs embedding models from an unlabeled embedding corpus using unsupervised techniques, including GloVe, FastText, and Word2Vec, as detailed in Chapter 4. The GloVe embeddings, chosen for their superior intrinsic evaluation performance, stand out among the available options (i.e., Section 6.4.1). The feature representation module converts the author's text into a lower-dimensional numeric representa-

tion (2D), where rows correspond to word indices and columns denote feature values. The training module is responsible for constructing classifier models using labeled datasets, while the testing module assesses the classifier models' performance with unlabeled datasets. An illustrative depiction of this authorship classification framework, employing CNN as described by Kim [12], is presented in Figure 5.2.

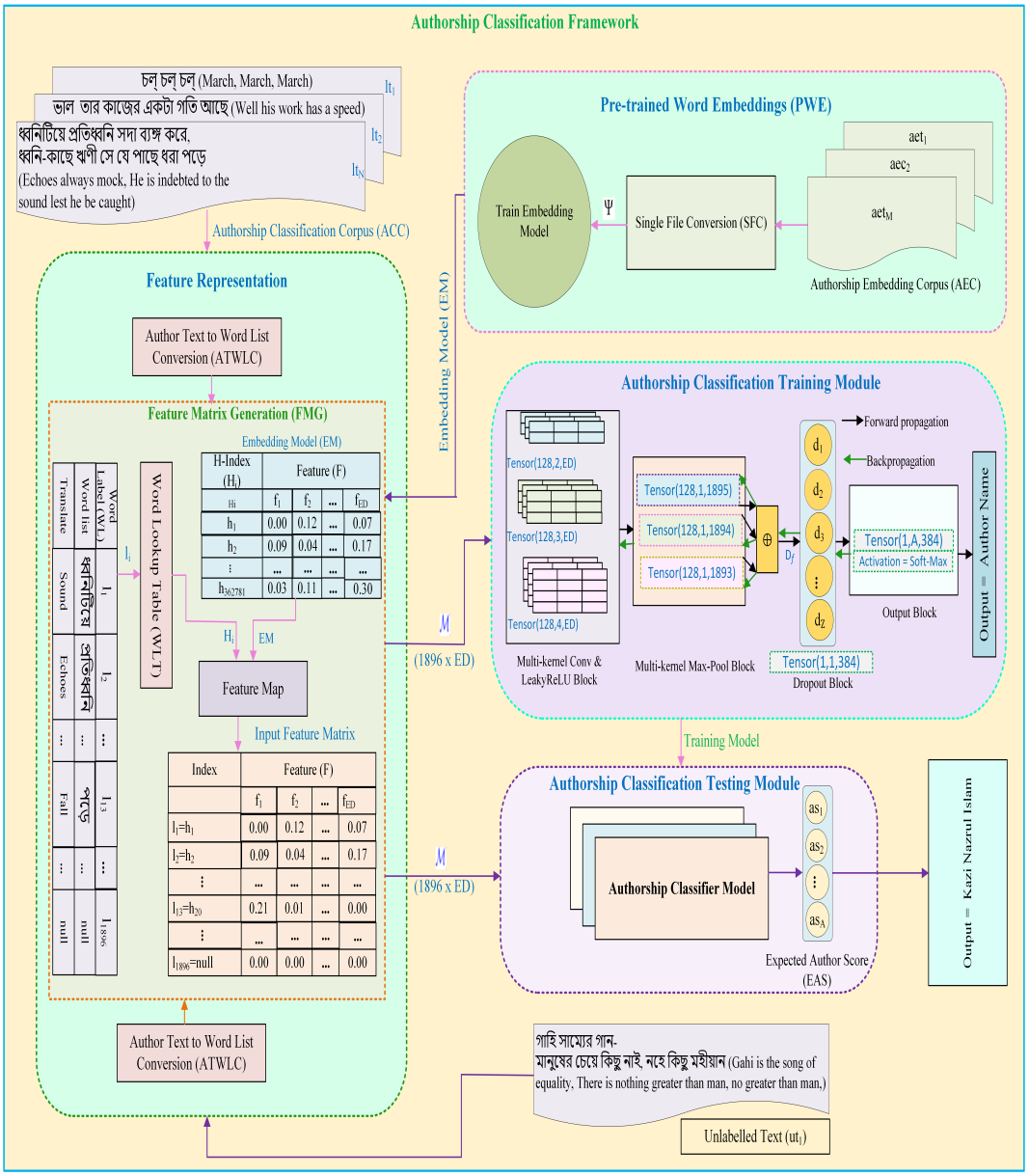


Figure 5.2: Proposed authorship classification framework

5.3.2.1 Feature Representation

In feature representation, the Authorship Classification Corpus (ACC) is utilized for training, while unlabeled text is used during testing. Features are represented

as an input feature matrix ($\mathcal{M}$). ACC is denoted as $\chi = lt_1, lt_2, lt_3, ..., lt_N$, where N represents the total number of texts in ACC. The Author Text to Word List Conversion (ATWLC) process transforms input lt_i into a list vector defined as $\mathcal{L} = l_1, l_2, l_3, ..., l_{\mathcal{W}}$, with $\mathcal{W}$ denoting the maximum number of words allowed in $\mathcal{L}$. In cases where the text contains fewer than 1896 words, the framework truncates the text to 1896 words and assigns null values.

Feature matrix generation (FMG) consists of two components: Word Lookup Table (WLT) and Feature Map (FM). WLT takes the i^{th} word l_i, where $l_i \in \mathcal{L}$, and generates a Hash Index (H_i). WLT consists of a $(key, value)$ vocabulary table, with "key" denoting the word (l_i) and "value" as the index (H_i). If a word is found in the vocabulary table, it returns the corresponding index (H_i); otherwise, it returns zero. Feature Map takes inputs (H_i and EM) and produces the output (input feature matrix ($\mathcal{M}$)). EM is represented as $\mathcal{G} = H_i \times F$, where $H_i = h_1, h_2, h_3, ..., h_{\mathcal{P}}$, $F = f_1, f_2, f_3, ..., f_{ED}$, and ED $\in 50, 100, 150, 200, 250, 300$. Here, h_i denotes the i^{th} vocabulary Hash-Index, and F represents the feature vector. $\mathcal{P}$ and ED indicate the total number of vocabulary and embedding dimensions, respectively. In the proposed framework, $\mathcal{P} = 362,781$, and ED takes values from $50, 100, 150, 200, 250, 300$. If $l_i \in H_i \times F$, the process extracts the corresponding feature from $\mathcal{G}$ and appends it in row-major order to IFM. If $l_i \notin H_i \times F$, the process adds ED zero values for l_i. As a result, the process generates $\mathcal{M}$ of size (1896× ED), where 1896 corresponds to the number of words, and ED represents the number of features in each word.

5.3.2.2 GloVe+CNN Training Module

The training module takes the feature matrix ($\mathcal{M}$) as input, where $\mathcal{M} \in (1896 \times$ ED), and produces a classifier model as output (Fig. 5.2). This module extracts stylometric and semantic features from multi-kernel CNN blocks at the author text level. Initially, the filter weights are initialized using Xavier initialization and updated using backpropagation. The Multi-kernel Conv and LeakyReLU Block uses the feature matrix $\mathcal{M} = (1896 \times \text{ED})$ to generate three different kernel tensors with shapes of $Tensor(128, 2, \text{ED})$, $Tensor(128, 3, \text{ED})$, and $Tensor(128, 4, \text{ED})$. These tensors are used for individual convolution operations. A special type of convolution produces a single-dimensional feature vector using Equation 5.1:

$$V_{[k^{th}:]} = \sum^{R} i = 1M[i : end] \otimes K[k^{th} : end] + Bi \tag{5.1}$$

Here, k^{th} kernel produces a $V_{[k^{th}:]}$ single-dimensional feature vector, and R denotes the input feature matrix rows. The bias value B_i is added to the out-

put convolution value. The output of the convolutions is $Tensor(128, 1, 1895)$, $Tensor(128, 1, 1894)$, and $Tensor(128, 1, 1893)$, respectively. The first value of the tensor, 128, represents the number of kernels. The Leaky ReLU operation is applied to all tensors. The convolution operation is followed by an activation operation within a block. The activation function serves as a data distribution re-scaling function, reducing overfitting and underfitting issues. In this architecture, the ReLU activation function is used (Equation 5.2).

$$V(i,j)^{k^{th}} = \begin{cases} a \times V(i,j)^{k^{th}} & \text{if } V(i,j)^{k^{th}} \leq 0 \\ V(i,j)^{k^{th}} & \text{otherwise} \end{cases} \tag{5.2}$$

In Equation 5.2, $V(i,j)^{k^{th}}$ represents the cell value of the i^{th} row and j^{th} column of the k^{th} kernel, with $V(i,j)^{k^{th}} \in \mathbb{R}^V$. The small multiplicand a is applied when the cell value is negative. Convolution and activation operations are followed by a pooling operation in the Multi-kernel Max-Pool Block. Pooling is a dimension reduction operation applied in the pooling layers. A max-pool operation is performed over the entire feature $V_{k^{th}}$ using Equation 5.3:

$$FP_{k^{th}}[1:128] = \arg\max_{j=1}^{j=128}(V_{k^{th}}[j:end]) \tag{5.3}$$

In Equation 5.3, k^{th} tensor generates a single-dimensional feature vector. In this architecture, three individual tensors produce three single-dimensional feature vectors. These feature vectors are concatenated using Equation 5.4:

$$D_f[1:N_d] = FP_1[1:128] \oplus FP_2[1:128] \oplus FP_3[1:128] \tag{5.4}$$

Where N_d represents the concatenated dimension with 384 feature values and 128 kernels.The dropout operation randomly sets some nodes to zero, controlling overfitting [135]. It is applied using Equation 5.5:

$$D_f[1:N_d] = (F[1:N_d] \otimes Dr[1:N_d]) + B[1:N_d] \tag{5.5}$$

In Equation 5.5, $D_f[1:N_d]$ follows a Bernoulli distribution. Each filter computes 384 values and drops out some node values based on the dropout threshold. The output block is the final part of the classification model, utilizing a tensor with a shape of $Tensor(18, 383)$ and a Softmax activation function. The tensor's internal values are 18 for the number of authors and 383 for feature values. The block contains a classifier matrix Θ (Softmax layer output), with rows representing author identity and columns representing trainable weights. The expected author

for any input can be calculated using Equation 5.6:

$$EX(Author = a|\mathcal{Z}) = \frac{(e^{\mathcal{Z}^T\Theta_a})}{\sum_{i=1}^{A} e^{\mathcal{Z}^T\Theta_i}} \tag{5.6}$$

where, $\mathcal{Z}$ is the author's feature value (i.e. 384), A is the total number of authors (i.e. 18) and Θ_a is the ath author feature values. Thus, the probable author is computed from the maximum expected value ($Max(EX(Author = a|\mathcal{Z}))$).

In a forward pass, the deviation or error between the input and output is calculated from the maximum expected value ($1 - Max(EX(Author = a|\mathcal{Z}))$). The error value is adjusted using the backpropagation technique [136], and each of the kernels is trained with adjustable values. In this way, the whole datasets pass through the networks 200 times and well-tuned the hyperparameter's value. At the end of the training, the model file is saved for the testing purpose. The layer-wise weight values are stored through the metafile. To investigate the effect of authorship classification performance, LSTM [13], Char-level-CNN [99], SVM [19], SGD [137], Multilingual pre-trained BERT (M-BERT) [16] and Distil-BERT [26] classifiers are also implemented on the same datasets.

5.3.2.3 GloVe+CNN Testing Module

This module takes the feature matrix ($\mathcal{M}$) as input, where $\mathcal{M} \in (1896 \times \text{ED})$, and generates an Author Expected Score (AES) vector as output. During testing, the feature representation process begins with an unlabeled text (ut_1) and produces an input feature matrix ($\mathcal{M}$). The feature matrix generation process remains the same as described in Section 5.3.2.1. If ut_1 contains fewer than 1896 words, zero padding is added. If it contains more than 1896 words, the text is truncated to 1896 words.

The generated feature matrix ($\mathcal{M}$) is passed through the authorship classifier model using trained kernel weights. The proposed architecture uses a dropout value of 0.5, and the test module generates a feature vector of dimension 385. This feature vector is projected into an output block weight matrix with a shape of 18×384. The Softmax function calculates AES using Equation (5.6). AES is defined as $\mathcal{A} = as_1, as_2, as_3, ..., as_{18}$ and identifies the maximum expected value (Max($\mathcal{A}$)) along with the corresponding index, indicating the predicted author's name. For instance, the authorship classifier model may produce the author name (i.e., class) as "Kazi Nazrul Islam" for an unlabeled input text (ut_1). *The comprehensive performance of the GloVe with CNN model and a comparison of its performance with other models will be discussed in Section 6.5.1.*

5.3.3 GloVe with VDCNN-based Model: Bengali Text Classification

The VDCNN system architecture consists of five core modules: Embedding model generation, two text-to-feature extraction modules, VDCNN training module, and VDCNN testing module. The overall document categorization system is depicted in Figure 5.3. In the embedding model generation phase, the system creates an embedding model, which is subsequently utilized by both the training and testing text representation modules to extract features. The VDCNN training module constructs the classifier model by leveraging the features of the training data, while the testing module employs the classifier model and features from unlabeled test samples to determine the document category. Figure 5.3 provides a detailed breakdown of the architecture of the VDCNN-based document categorization system.

The embedding model generation is described in Chapter 4 and the GloVe model is selected based on intrinsic evaluation performance, and the remaining three subsequent paragraphs are described in the following.

5.3.3.1 Text to Feature Extraction

Two feature extraction modules are employed in the system: one for labeled input text and the other for unlabeled input text. Each module comprises Labeled/Unlabeled input text, Text List Conversion (TLC), List (L), Embedding Model (EM), and Feature Extraction (FE). The labeled input text is used in the VDCNN training module, while the unlabeled input text is employed in the VDCNN testing module. Both training and testing text feature representation follows a similar procedure, starting with the input text document and resulting in a 2D feature matrix for use in the respective module. The feature extraction process involves three phases: Text to text-to-list conversion (TLC), GloVe model generation, and Feature Extraction (FE). TLC takes the input text document (td_i) and converts it into a word list vector (L), consisting of words represented as $L = [l_1, l_2, l_3, ..., l_N]$, where l_i corresponds to the i^{th} word and N is the maximum length set at $2,048$. Input texts exceeding this length are truncated to the first $2,048$ words, with zero padding applied to texts containing fewer words.

In the FE process, the list vector (L) and an Embedding Model (EM) are used as inputs. EM is a 2D vector with $1,517,390$ unique words, each assigned an individual h-index and corresponding feature values denoted by ED $\in \{25, 50, 100, 150, 200, 250, 275, 300, 325, 350, 400\}$. FE maps ($H_i$) to EM to extract features. If the (L) word (H_i) is found in EM, FE returns the corresponding

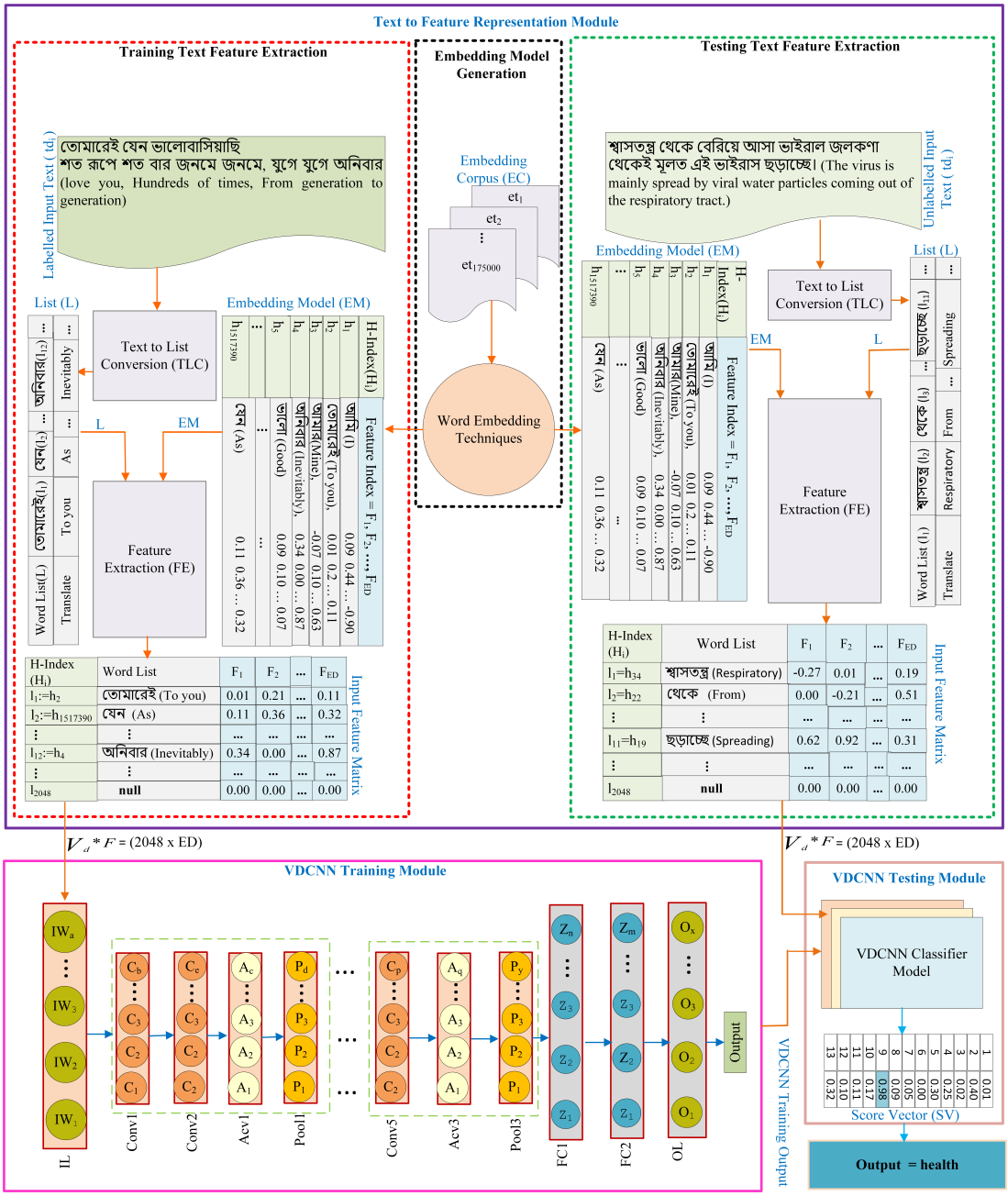


Figure 5.3: Proposed very deep convolution neural network-based text categorization

(H_i) feature values; otherwise, it returns a null vector for ED features. For instance, the first word (l_1) is mapped to the index h_2 of EM. The FE phase generates an output feature matrix $(V_d * F)$, where V_d represents the number of words and F represents the number of features. Each word (l_i) in a text document is arranged in rows, with corresponding features in columns, resulting in a feature matrix with dimensions of $(2048 \times ED)$. This matrix is an H-Index (H_i) representation, where each row is a feature index, and each column is a feature. The process of generating the 2D feature matrix for testing text follows the same

technique used for training. The upper right part of Figure 5.3 illustrates the process of testing text feature extraction.

5.3.3.2 GloVe+VDCNN Training Module

In the training module, a document classifier model is built using Bengali corpora. The input for the VDCNN consists of the 2D feature matrix ($V_d * F$), which traverses through various layers including the input layer, convolution layer, pooling layer, activation layer, and fully connected layer. The architecture of the VDCNN training module is visualized in Figure 5.4.

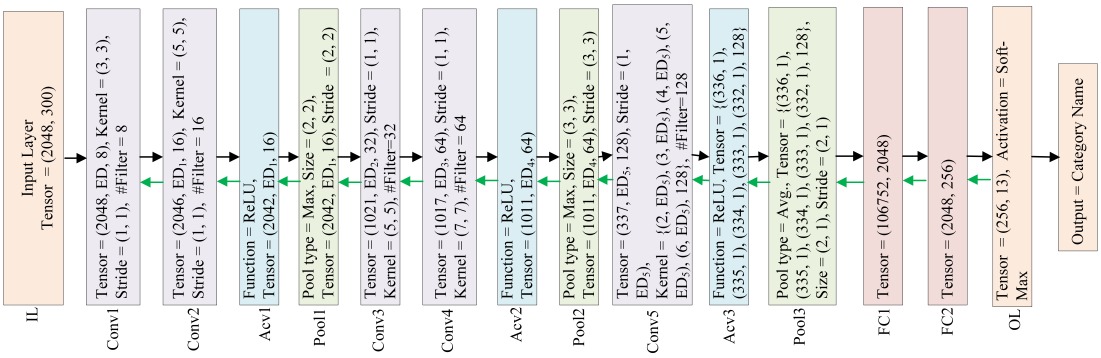


Figure 5.4: Proposed VDCNN classifier architecture. A left green arrow indicates the backpropagation, and right black arrow denotes the forward propagation

The Input Layer (IL) consists of a set of tensor nodes, represented as $IL = IW_1, IW_2, IW_3, ..., IW_a$, with a corresponding to the tensor length of 2048. This layer takes the initial feature vector from the word embedding model and forwards it to subsequent VDCNN layers for further processing. The size of the input layer tensor is $Tensor(2048, ED)$, where ED refers to the embedding dimension. In the following sections, the tensor format is presented as $Tensor(heights, width, filter)$, replacing height and width with the number of words and feature length and filter with the number of filters. The convolution layers in the architecture, denoted as $Conv1, Conv2$, ..., $Conv5$, act as local feature extractors with fine-tuned kernels using backpropagation [138]. Conv1, for instance, consists of tensor nodes $(C_1, C_2, C_3, ..., C_b)$, where C signifies the tensor nodes, and the subscript (b) indicates the tensor height ($b = 2048$ in Conv1). The i^{th} layer input is propagated to the output according to Eq. 5.7.

$$\tau^i = \tau^{i-1}. \tag{5.7}$$

Here, τ^i represents the kernel/weight metric for the i^{th} layer. During convolution, each kernel element-wise multiplies with the input matrix. The coordinate posi-

tion, for instance, $input(s, r)$, is updated using Eq. 5.8, where s and r represent the coordinate position in the 2D input matrix.

$$input(s,r)^i = \sum_{p=0}^{k_h^i} \sum_{q=0}^{k_w^i} input(p+s, q+r) \times K(p,q) \tag{5.8}$$

In this equation, k_h^i and k_w^i denote the height and width of the kernel for the i^{th} layer. p and q are the kernel coordinates, and K(p,q) slides through the input matrix. This convolution operation extracts local (e.g., sentence-level) and global (e.g., document-level) features based on document semantics. Smaller kernel sizes capture local details, while larger ones extract abstract document views. In each convolution layer, the number of outputs matches the number of kernels, and all parameters are broadcasted to subsequent layers. The tensor dimensions for the five convolution layers ($Conv1$ to $Conv5$) are Tensor(2048, ED, 8), Tensor(2046, ED_1, 16), Tensor(1021, ED_2, 32), Tensor(1017, ED_3, 64), and Tensor (337, ED_5, 128), where ED to ED_5 vary based on the layer operations.

In the activation layer, the nodes determine whether to transmit their values to the next layer based on the ReLU activation function, which speeds up training [139]. This operation is defined as follows (Eq. 5.9):

$$input(s,r)^i = max(input(s,r)^{i-1}, 0) \tag{5.9}$$

After activation, the non-zero nodes proceed to the next layer. Three activation layers ($Acv1$, $Acv2$, and $Acv3$) have tensor shapes of $Tensor(2042, ED_1, 16)$, $Tensor(1011, ED_4, 64)$, and $Tensor(336, 1)$, $Tensor(335, 1)$, ..., $Tensor(332, 1)$, 128), where A represents activation tensor nodes, and c is the tensor height. The pooling operation reduces the input dimension and generalizes the feature map. Three pooling layers ($Pool1$, $Pool2$, and $Pool3$) employ max-pooling as the pooling function. The architecture features two fully connected layers (FC_1 and FC_2). The convolution layer's high-level features are flattened in the fully connected layer. FC_1 contains 106,752 neurons connected with 2,048 neurons in FC_2. A dropout operation is applied to mitigate overfitting, and the hierarchical structure of weighted kernels is established. The output layer takes a 1D feature map ($Tensor(256, 1)$) as input and outputs an expected category. The Soft-Max function normalizes the class values, determining the expected class name. The model trains using forward and backward propagation until convergence or minimal error is achieved. The study compares this model's classification performance with other models using the same datasets, summarized in Table 6.14.

5.3.3.3 GloVe+VDCNN Testing Module

The testing module's purpose is to assign an expected class to an unlabeled text document (tdi) from a predefined set of classes. This module utilizes the result of the VDCNN training model and the output of the testing text feature extraction module, which provides an input feature matrix ($Vd * F$). The VDCNN testing layers load the trained metafiles and apply kernel weights to the feature matrix, resulting in a 2D score vector (SV). This vector contains classification scores, $s_1, s_2, s_3, ..., s_{CL}$, where CL represents the number of classes. The scores are computed as per Eq. 5.10:

$$s_i = \frac{e^{s_i}}{\sum_{j=1}^{CL} e^{s_j}} \tag{5.10}$$

The maximum expected value within the vector determines the corresponding class, which becomes the final output class. For instance, in Fig. 5.3, the VDCNN classifier model generated a maximum expected value of 0.98 at index 9, resulting in the class assignment of "health. *The comprehensive performance of the GloVe with VDCNN model and a comparison of its performance with other models will be discussed in Section 6.5.2.*

5.3.4 Meta-Embedding with CNN-based Model: Bengali Text Classification

This study aims to create an intelligent Bengali text classification system, known as Meta-Embedding with CNN (AVG-M+CNN), which leverages meta-embedding and CNN for accurate categorization of Bengali text into predefined classes. The proposed system comprises four crucial modules: (i) Text to Feature Extraction (AVG-M), (ii) Automatic Hyperparameters Tuning and Selection (AHTS) for CNN, (iii) CNN Training, and (iv) CNN Testing. An abstract representation of AVG-M+CNN is illustrated in Fig. 5.5.

5.3.4.1 Average Meta-Embedding (AVG-M)

The AVG-M module takes three inputs: the GloVe model denoted as $E_{B[1]}$, the FastText model denoted as $E_{B[2]}$, and a set of corpora {BARD, IndicNLP, Prothom-Alo, BTCC11}. It produces two average feature fusion matrices: MF_{ij}^{l} for the i^{th} corpus and j^{th} training sample, and MF_{ij}^{u} for the i^{th} corpus and j^{th} testing sample. In this context, $|V_k|$ represents the total number of words, and ED indicates the feature dimension. To evaluate the performance of AVG-M+CNN, four distinct classification corpora are utilized, namely BARD, Indic-

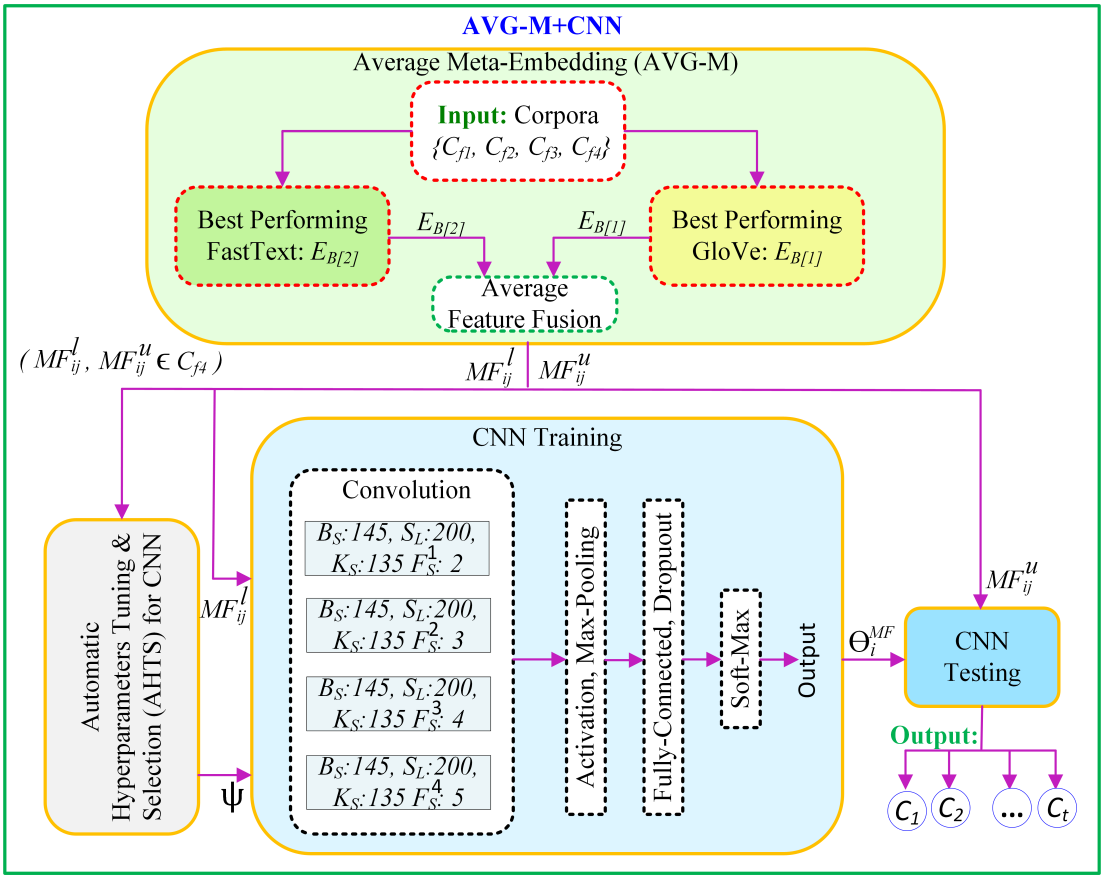


Figure 5.5: Overview of the proposed text classification model (AVG-M+CNN)

NLP, Prothom-Alo, and BTCC11. Each of these corpora, represented by C_{f_i}, is randomly divided into a 70:30 ratio, with 70% of the texts allocated for training and 30% for testing purposes.

5.3.4.2 AVG-M+CNN Training

In the CNN Training module, we take as inputs the tuned CNN hyperparameters denoted as Ψ, and the i^{th} corpus j^{th} sample feature matrix, represented as MF^l_{ij}. These inputs lead to the creation of classification models Θ^{MF}_i.

The i^{th} feature matrix, MF^l_{ij}, is then passed through the CNN input layer. Four distinct filters are employed to extract semantic and syntactic features. The convolution operation is performed on MF^l_{ij} with the q^{th} filter, F^q_S, using the following equation Eq. 5.11:

$$C^q_O[1 : (|V_k| - |F^q_S| + 1)] = MF^l_{ij} \otimes W^q_t + bias^q \tag{5.11}$$

Here, C^q_O represents the convolution output of the q^{th} operation, and $|V_k|$ is the maximum input sequence length. The trainable weighted values of the q^{th} filter

are given by W_t^q. The single-layer multi-kernel CNN operations are followed by an activation layer where the LeakyReLU operation is applied element-wise to control overfitting and underfitting. The i^{th} output of the LeakyReLU operation is computed as follows Eq. 5.12

$$C_O^i[i:q]q = 1^{|CO^i|} = \left\{ \begin{matrix} C_O^i[i{:}1] \ \ if \ \ C_O^i[i{:}1] \geq 0 \\ \\ NS \times C_O^i[i{:}1] \ \ otherwise \end{matrix} \right\} \tag{5.12}$$

Here, NS represents the negative slope, and a default value (e.g., $1e-2$) is utilized. After the activation layer, a filter-wise max-pooling operation is performed on the output, and the resulting max-pool features are concatenated as shown below:

$$\tau = \left\{ \tau \oplus C_O[i:end] \right\}_{i=1}^{i=\|C_O\|} \tag{5.13}$$

Here, τ signifies the concatenated feature vector, and in this context, its length is 540 (e.g., four filters & a maximum of 135 filters).

A dropout operation is applied to the concatenated features to address overfitting. The dropout vector (D_v) is generated with the Bernoulli distribution and element-wise multiplication is carried out with the concatenated feature vector, i.e., $\tau \otimes D_v$. The output of this multiplication is then forwarded to the softmax layer, which calculates the i^{th} sample's expected value using the following equation:

$$output^i = \frac{\exp^{(\Theta_W[i] \otimes \tau^i)}}{\exp^{(\sum_{q=0}^{C_t}(\Theta_W[q] \otimes \tau^i))}} \tag{5.14}$$

In this equation, Θ_W represents the softmax layer weight matrix, C_t is the total number of categories, and $|\tau|$ denotes the softmax layer input feature length (e.g., 540). This process calculates all the expected values and selects the category corresponding to the maximum expected value.

5.3.4.3 AVG-M+CNN Testing

The AVG-M+CNN classification framework is trained using 70% of the data from the corpus C_f, and its performance is evaluated on the remaining 30% test data. These test samples generate feature matrices, denoted as $MF_{ij}^u \in \text{I!R}^{|V_k^i| \times ED^i}$, using the feature extraction module. These matrices are then fed into the AVG-M+CNN test module. The AVG-M+CNN module assigns classes to the test data using Eq. 5.14, and the test evaluation system assesses the performance of this

assigned labeling. The four benchmark corpora, specifically BARD, IndicNLP, Prothom-Alo, and BTCC11, are evaluated using the AVG-M+CNN frameworks. *The comprehensive performance of the Meta-Embedding with CNN model and a comparison of its performance with other models will be discussed in Section* 6.5.3.

5.3.5 Attention-based Meta-Embedding and ACNN-based Model: Bengali Covid Text Classification

The CovTiNet, which stands for the Proposed Covid Text Identification System, consists of two essential modules: the Attention-based Position Embedding Averaging of GloVe and FastText (APeAGF) for text feature representation and the Attention-based Convolutional Neural Networks (ACNN) for Covid text identification. Figure 5.6 provides an abstract overview of the CovTiNet. The subsequent subsection will delve into the specifics of these two modules.

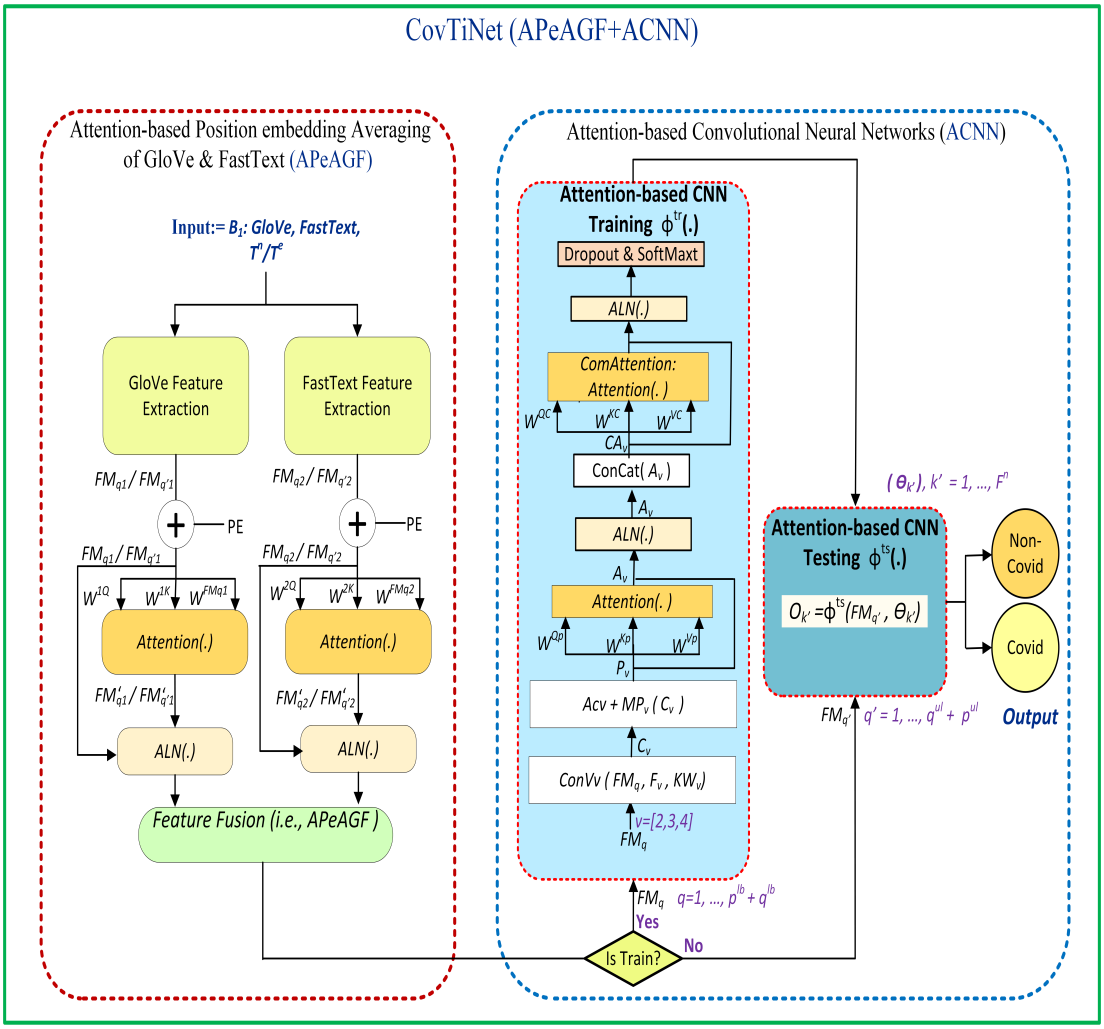


Figure 5.6: High-level View of CovTiNet

5.3.5.1 Attention-based Text to Feature Extraction (APeAGF)

In Figure 5.6, the attention-based position embedding averaging of GloVe and FastText (APeAGF) module takes input as training and testing set, i.e., $(T^n/T^e) \in BCovC$ and output is the feature matrix (e.g., $FM_q/FM_{q'}$). The $q^{th} \in T^n$ training and $q^{'th} \in T^e$ testing sample is sequentially represented the features matrix FM_{q1} and $FM_{q'1}$ for GloVe embedding, whereas FM_{q2} and $FM_{q'2}$ for FastText embedding using Eq. 5.15.

$$PE_{q/q'}[1:sl] = \begin{cases} sin(\frac{pos}{10000^{(2\times pos/ED)}}), \ if \ \ (pos\%2)==0 \ \ pos=1,...,sl \\ \\ cos\frac{pos}{10000^{(2\times pos+1/ED)}}, \ otherwise \ \ pos=1,...,sl \end{cases} \tag{5.15}$$

Here, ED denotes the embedding dimension, and the word position of $q/q^{'}$ sample is $pos \in sl$. This position encoding is just added to the training/testing sample $(q/q^{'})$ and fed to the trainable self-attention block, i.e., $Attention(.)$. The self-attention block contains nine trainable weight matrices, e.g., three for GloVe, three for FastText, and three for Word2Vec. The generalized form of matrices are query $(W^{aQ} \in \mathbb{R}^{sl\times ED})$, keys $(W^{aK} \in \mathbb{R}^{sl\times ED})$ and values $(W^{FMqa} \in \mathbb{R}^{sl\times ED})$ for GloVe, FastText and Word2Vec respectively.

In addition to better syntactic feature representation, position encoding (PE) is added to these feature matrices. The function of $Attention(.)$ calculates the attention value of each word in the feature matrix and improves the contextual representation of training/testing samples (i.e., $q/q^{'}$) using Eq. 5.16.

$$FM^{'}_{qa} = (\frac{(FM_{qa} \times W^{aQ}) \times (FM_{qa} \times W^{aK})^T}{\sqrt{ED}}) \times (FM_{qa} \times W^{FMqa}) \tag{5.16}$$

Here, FM_{qa} indicates the input fused feature matrix whereas $FM^{'}_{qa}$ represent the attention-based output fused feature matrix of a^{th} best performing embedding model (i.e., $a = [1 : GloVe, 2 : FastText, 3 : Word2Vec]$). The addition and layer normalization block is combined with an attention-based and positional encoding feature, which improves the word-level correlation [132].

The attention value normalization functions $ALN(.)$ take the attention-based feature matrix and original feature matrix (i.e., take from skip-connection). The $ALN(.)$ function normalized the attention value and forwarded it to the feature fusion module. The feature fusion module is just averaging the attention-based feature matrix of GloVe and FastText. Finally, the APeAGF module output FM_q for q^{th} training sample attention-based feature matrix and $FM_{q'^{th}}$ testing sample attention-based feature matrix. The FM_q will be used for training purposes, and

$FM_{q'}$ will be used for testing purposes.

Attention-based CNN: ACNN The Attention-based Convolutional Neural Networks (ACNN) module works in two steps. The ACNN module training with the training set T^n generates a Covid text identification model in the first step. In the second step, the Covid text identification model is evaluated by the testing set T^e and calculates the performance of the ACNN module. The following paragraphs describe the details of the two steps.

5.3.5.2 Attention-based CNN Training

The training function $\Psi(.)^{tr}$ takes the training samples fused feature matrices $FM_q : FM_m \in \mathbb{R}^{sl \times ED}$ and outputs a Covid text identification model $\Theta_{k'}$. Initially, a convolution operation is applied to the single CNN layer with three different kernel sizes (i.e., $v = [2, 3, 4]$). The v^{th} kernel conducted the convolution operation ($ConV_v$) using Eq. 5.17.

$$C_v[1 : F_v] = KW_v[v : ED] \otimes FM_q + bias_v, v = [2, 3, 4]; q = 1, ..., p^{lb} + q^{lb} \quad (5.17)$$

Here, v^{th} trainable kernel indicates KW_v and the convolution output is represented by C_v. The three kernels' convolution output is stored in C_v and is forwarded to the second layer. The second layer applies kernel-wise activation and max-pooling operations. The v^{th} kernel activation and max-pooling operation is conducted by Eq. 5.18.

$$P_v[1 : len(F_v)] = MaxPooling(ReLU(C_v[1 : F_1])), v = [2, 3, 4] \quad (5.18)$$

The $ReLU(.)$ activation function ($AC_v(.)$) normalized the sentence-level convoluted features and the max-pooling function $MP_v(.)$ returns a single maximum value from the trainable convolution output (i.e., C_v). The output of the max-pooling operation is stored in P_v and forwarded to the third layer (i.e., $Attention(.)$). The attention layer calculated the sentence-level attention using Eq. 5.16 and concatenated the three kernels (e.g., $v = [2, 3, 4]$) attention-based feature (A_v). This concatenated feature is passed to another attention-based encoding layer ($ComAttention$), and the dropout operation is applied. The dropout operation randomly blocks some neuron values, which helps overcome the overfitting issues. Finally, the dropout features are forwarded to the softmax layer, predicting the Covid identification score. The error value is calculated from the

predicted and ground-truth value and adjusts the error using the backpropagation operation. At the end of the training, the attention-based CNN saves a Covid text identification model ($\Theta_{k'}$), which is used in the next phase (i.e., the attention-based CNN testing phase).

5.3.5.3 Attention-based CNN Testing

The attention-based CNN test function calculates the model's ability to perform the task ($\Psi^{ts}(.)$). The function takes the Covid text identification model ($\Theta_{k'}$) and sequentially predicts the test set samples (T^e). The $q^{'}$ test sample fused feature matrix $FM_{q'} : FM_{q'} \in \mathbb{R}^{sl \times ED}$. The fused feature matrix is fed to the pre-trained mode ($\Theta_{k'}$) and calculates the expected value using Eq. 5.19.

$$E_{k'}[q^{'}] = \Theta_{k'} \times FM_{q'}, q^{'} = 1, ..., p^{ul} + q^{ul} \tag{5.19}$$

The $E_{k'}[q^{'}]$ denotes the expected value of $q^{'th}$ test sample (i.e., T^e). Now, the expected value is normalized by Eq. 5.20.

$$O_{k'}[q^{'}] = max(\frac{e^{(E_{k'}[q^{'}])}}{\sum_{z=1}^{z=p^{ul}+q^{ul}} e^{(E_{k'}[z])}}), q^{'} = 1, ..., p^{ul} + q^{ul} \tag{5.20}$$

Here, $O_{k'}[q^{'}]$ indicates normalized expected value of $q^{'}$ test sample. All of the statistical measures will use this outcome of ($O_{k'}[q^{'}]$) to evaluate the performance of the model. *The comprehensive performance of the APeAGF with ACNN model and a comparison of its performance with other models will be discussed in Section 6.5.4.*

Drawbacks of Deep Learning-based Classifiers: Deep learning-based classifiers, including Convolutional Neural Networks (CNN), Very Deep Convolutional Neural Networks (VDCNN), Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM), and Gated Recurrent Unit (GRU), demonstrate robust text classification capabilities. However, when applied to low-resource Bengali text, these models encounter limitations. Despite achieving reasonable accuracy, they still possess several drawbacks, which will be detailed below:

- Exhibit a high dependency on data availability, rendering them less effective with limited annotated datasets, and are susceptible to overfitting, a critical concern in scenarios with small data.
- Computational demands can strain resource-constrained environments, and the intricate process of hyperparameter tuning is often a time-consuming

task.

- Effectiveness of these classifiers is influenced by the quality and coverage of word embeddings, which can be limited for Bengali text language

To address these issues, alternative techniques such as *attention-based early fusion* models or *transfer learning with transformer-based models* may be more suitable for low-resource Bengali text classification, providing improved performance and mitigating these drawbacks.

5.4 Transformer-based Text Classifiers

To address challenges related to non-contextual embeddings, such as out-of-vocabulary (OOV) words and the lack of contextual features, this study employs transformer-based language models. Traditional deep-learning text classification models often struggle to capture both local and global semantics effectively, leading to reduced performance in tasks involving short texts and morphologically rich languages. In response, transformer-based language models are fine-tuned using Bengali text classification corpora to overcome these limitations in deep learning-based text classification. Considering the Bengali linguistic variations, it is essential to tune the hyperparameters of transformer-based models before conducting the fine-tuning operation. This tuning process is achieved through the following method.

5.4.1 Models Hyperparameters Tuning

Hyperparameter tuning for Transformer-based models in low-resource languages is vital for optimizing their performance. These models often require fine-tuning to adapt to specific language characteristics. Key hyperparameters such as batch size, sequence length, and learning rate are crucial in achieving optimal results. Fine-tuning customizes these models for low-resource languages, enhancing their effectiveness in NLP tasks. Our research introduces a dedicated hyperparameter tuning algorithm to address this challenge. It's worth noting that while pre-trained language models have achieved state-of-the-art results, their performance depends significantly on hyperparameters. This study focuses on the impact of three influential hyperparameters—epoch number, batch size, and maximum sequence length—on fine-tuning pre-trained language models for low-resource authorship attribution, such as Bengali text classification and authorship attribution, within hardware and time constraints. The detailed workings of our hyperparameter tuning algorithm can be found in Algorithm 10.

Algorithm 10: Automatic Hyperparameters Tuning and Selection

$\mathcal{H}_l^I : \{\}, \mathcal{H}_l^t : \{\}, best_{accuracy} := 0$
$\mathcal{H}_l^I[1] := \{1, 2, ...15\}$ //number of epoch
$\mathcal{H}_l^I[2] := \{2, 3, ...12\}$ // batch size
$\mathcal{H}_l^I[3] := \{50, 60, ...750\}$ // maximum sequence length
$Function : \lambda(C_i, TL_l(\mathcal{H}_l^I))$ // Input Function {
for *e in range(1, len*$(\mathcal{H}_l^I[1])$ **do**
  for *b in range(1, len*$(\mathcal{H}_l^I[2])$ **do**
    for *s in range(1, len*$(\mathcal{H}_l^I[3])$ **do**
      ep:=$\mathcal{H}_l^I[1][e]$, bs:=$\mathcal{H}_l^I[2][b]$, sl:=$\mathcal{H}_l^I[3][s]$
      τ_{ac}:=$\Phi_l^{FT}(ep, bs, sl, C_i)$ // Fine-tuned for corpus C_i// **if** $(best_{accuracy} < \tau_{ac})$ **then**
        $best_{accuracy} := \tau_{ac}$ // Updated best accuracy
        $\mathcal{H}_l^t[2] = sl$
    end
  end
end
return (H_l^t) // Tuned hyperparameters }

From Algorithm 10, hyperparameters tuning function $\lambda(.,)$ takes two inputs: first input is hyperparameters of transformer-based language model $TL_l(H_l^I)$ and the second one is corpus C_i. The $TL_l, l \in$ {M-BERT, Bengali-BERT, M-Distil-BERT, Bengali-ELECTRA, Indic-BERT, MuRIL, XML-Roberta, AraBERT} denotes the transformer-based language models and H_l^I denotes the initial hyperparameters set. Whereas $C_i, i \in \{TextClassificationCorpora\}$ indicates the corpus set. The output of this module is the tuned hyperparameters, i.e., $TL_l(H_l^t)$. These tuned hyperparameters are forwarded to the authorship attribution training module.

5.4.2 Fine-tuning for Arabic Covid Text Classification

This research introduces a system for identifying Arabic COVID-19 text, which comprises three core modules: (i) Arabic Text to Feature Extraction, (ii) Transformer-based Fine-tune Training, and (iii) Transformer-based Fine-tune Testing. An overview of the proposed system is depicted in Figure 5.7. The forthcoming subsections will provide detailed explanations of each module.

5.4.2.1 Arabic Text to Feature Extraction

The j^{th} text sample (i.e., S_j) of the AraCoV corpus is represented as $S_j = \{w1, w2, ..., wL\}$, where $w1$ denotes the first word, and wL denotes the last word.

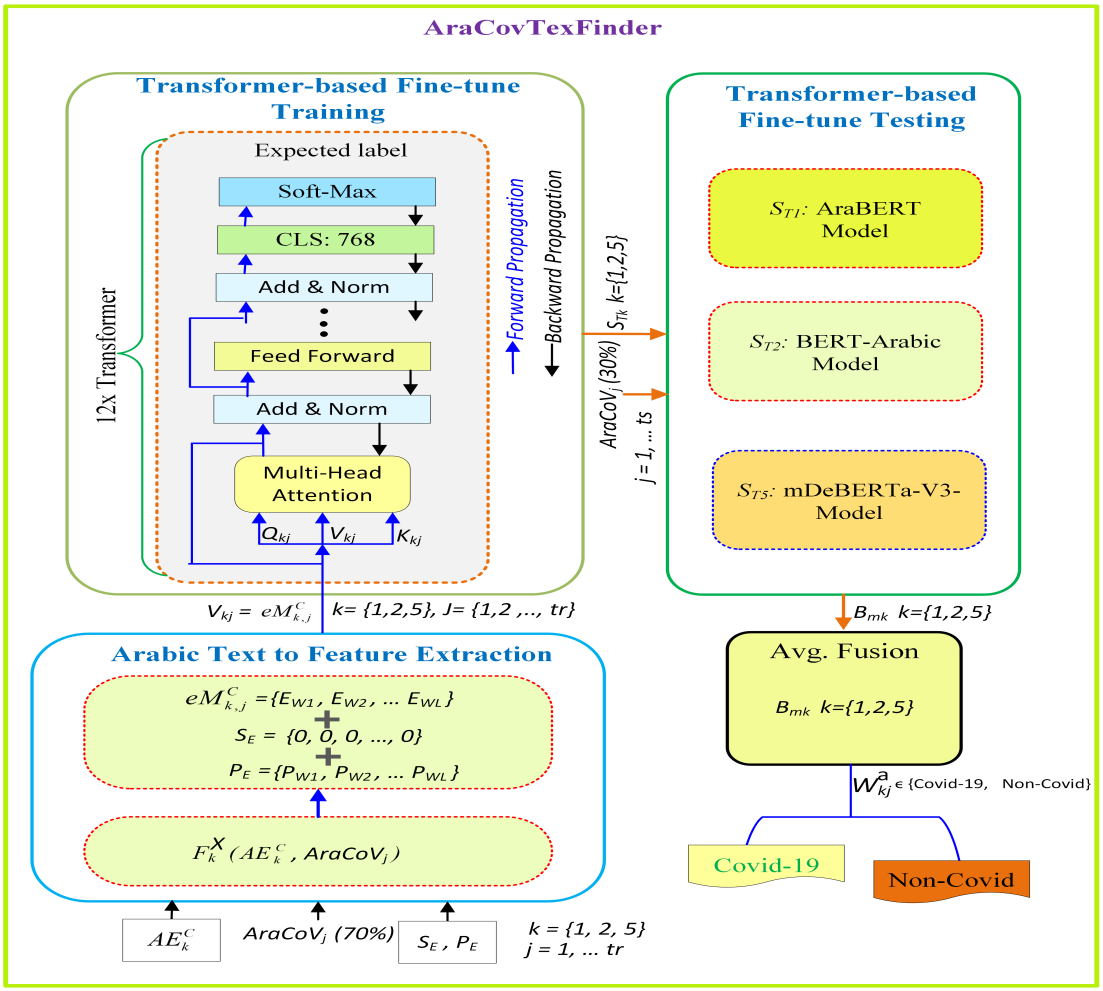


Figure 5.7: Abstract View of Proposed System: AraCovTexFinder

Now, using Eq. 5.21.

$$eM^C_{k,j} = F^X_k(AE^C_k, AraCoV_j), j \in \{1, 2, ..., A_s\}, \\ k \in \{AraBERT, BERT - Arabic, ..., mDistilBERT\} \tag{5.21}$$

The k^{th} transformer-based embedding of j^{th} sample is producing a features matrix $eM^C_{k,j} \in \mathbb{R}^{(L \times F_d)}$ and $eM^C_{k,j} = T_E \oplus S_E \oplus P_E$. L indicates the maximum sequence length, F_d represents each word's feature dimension, and $\oplus$ indicates the element wise summation operator. T_E indicates the transformer-based token embedding and $T_E = \{E_{w1}, E_{w2}, ..., E_{wL}\}$. S_E denotes the segment embedding; this research segment value is zero. The P_E represent the position-based encoding and $P_E = sin(\frac{pos}{10000^{(2 \times pos/ED)}})$ for even position encoding as well as $P_E = cos(\frac{pos}{10000^{(2 \times pos+1/ED)}})$ for odd position encoding. This feature matrix is fed to the transformer-based fine-tuning training network.

Transformer-based Fine-tune Training Based on *ablation analysis* and *late fusion* formulation, two best-performed monolingual (i.e., AE_1^C: AraBERT & AE_2^C: BERT-Arabic) and one multi-lingual (i.e., AE_5^C:mDeBERTa-V2) transformer-based language models are selected for the fine-tuning purpose. The k^{th} transformer-based fine-tune training function ($\Psi_k^T(.)$) take the j^{th} sample feature matrix $eM_{k,j}^C$ and trained using Eq. 5.21. The training phase is initially fed to the *Multi-head Attention* block and calculates the self-attention for k^{th} transformer of j^{th} sample using Eq. 5.22.

$$A_{kj} = softmax(\frac{Q_{kj} \times K_{kj}^{'}}{\sqrt{F_d}}) \cdot eM_{k,j}^C, j \in \{1, 2, ..., tr\}, k = \{1, 2, 5\} \tag{5.22}$$

Where A_{kj} denotes the attention matrix. The Q_{kj} and $K_{kj}^{'}$ represents the trainable *query* and transposed *key* matrix. The F_d indicates the dimension of the feature. The 12 attention blocks calculate the sentence and document-level semantics and forward to the *Feed Forward* block. Finally, the 768 dimensions *CLS* features are fed to the *Soft-Max* block, and calculate the expected COVID-19 affiliated class value is calculated using Eq. 5.23.

$$Y_{kj} = softmax(\Omega_k \times (CLS)_j^{'} + B_{kj}), j \in \{1, 2, ..., tr\}, k = \{1, 2, 5\} \tag{5.23}$$

Where Y_{kj} indicates the j^{th} expected class value and Ω_k denotes the trainable weighted matrix of k^{th} transformer-based fine-tuned model. The *cross-entropy loss* is calculated from the Y_{kj} and updated all the trainable parameters using the *backward propagation* operations. In this way, all of the training samples of the AraCoV corpus are passed up to the loss value convergence phase, i.e., in ten epochs, the COVID-19 text identification model has obtained the optimal loss value for the AraCoV test samples. At the end of the training, the k^{th} transformer-based fine-tuned model S_{Tk} was forwarded to the test phase module.

5.4.2.2 Transformer-based Fine-tune Testing

In this module, input takes three fine-tune COVID-19 text identification models such as AE_k^C where $k = \{1, 2, 5\}$, AraCoV corpus (i.e., 30% AraCoV corpus) and output is a performance matrix. The j^{th} test sample is extracted feature matrix $eM_{k,j}^C$ using the k^{th} transformer-based embedding model and calculates the expected label using Eq. 5.24.

$$E_{kj} = softmax(\eta(eM_{k,j}^C, S_{Tk})), j \in \{1, 2, ...ts\}, k = \{1, 2, 5\} \tag{5.24}$$

Where $\eta(.,.)$ denotes the forward pass function and E_{nk} denotes the j^{th} sample *softmax* score. The ts represents the total number of test samples on the AraCoV corpus. The weighted average late fusion score of n^{th} test sample is calculated using Eq. 5.25.

$$W_{kj}^{a} = \frac{\sum_{k=\{1,2,5\}}(E_{kj})}{3}, k = \{1,2,5\}, j \in \{1,2,...ts\} \tag{5.25}$$

Where W_{kj}^{a} denotes the j^{th} test sample weighted average late fusion score and $W_{kj}^{a} \in$ {Covid-19, Non-Covid}. All performances are calculated from the W_{kj}^{a} and ground truth test samples. *The comprehensive performance of the AraCov-TexFinder model and a comparison of its performance with other transformer-based models will be discussed in Section* 6.5.5.1.

Constrained of Transformer-based Classifiers While transformer-based language models have achieved state-of-the-art performance in the resource-constrained text classification domain, they still have some constrained.The constrained will be detailed bellow:

- Resource-constrained languages often face challenges with the default hyperparameters of high-resource language models, leading to suboptimal performance.
- The absence of generalization in both multilingual and monolingual language models is a noteworthy concern.
- Restricted ability to identify crucial features for a specific classification mode.

To address these issues, this research introduces an attention-based early fusion technique that combines the features of monolingual and multilingual language models, leading to improved classification performance. In the following sections, we will provide a detailed description of this proposed attention-based early fusion system for resource-constrained text classification. The system's efficacy is evaluated using five text classification corpora (BARD, OSBC, BTCC11, IndicNLP, and Prothom-Alo) and further assessed with four authorship attribution corpora (BACC-18, BAAD16, BAAD16, and LD) for a comprehensive analysis.

5.4.3 Proposed Attention-based Early Fusion Network: AFuNeT

The goal of this research is to develop an *intelligent* text classification system for low-resource languages called AFuNet that can *intelligently* identify a piece of text category based on text semantics. The development process of AFuNet involves four phases: (i) Baseline Models Evaluation and Hyperparameters Tuning, (ii) Late Fusion and Selection of the Best-performing Models, (iii) Attention-based Early Fusion and Identification of the Best-performing Model, and (iv) Ablation Study and Impact Analysis. Figure 5.8 illustrates an abstract process of the AFuNet framework.

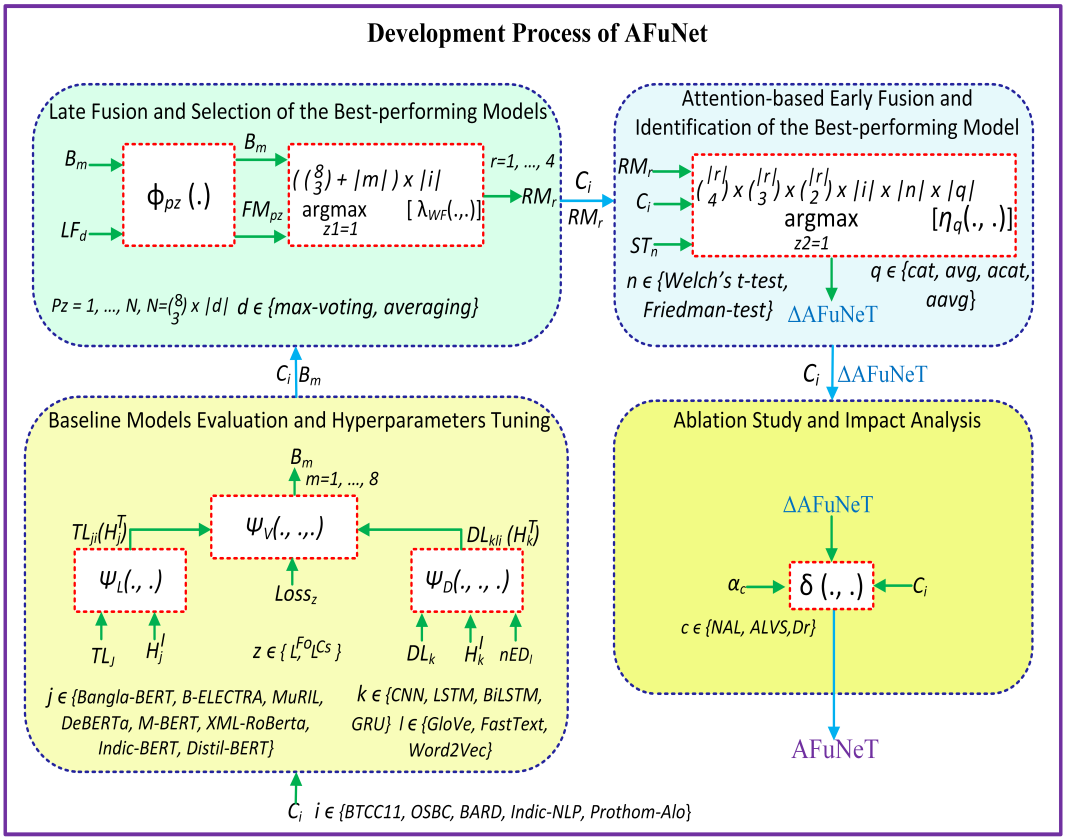


Figure 5.8: Abstract process of the AFuNet framework

5.4.3.1 Baseline Models Evaluation and Hyperparameters Tuning

To address **RQ2**, the Transfer Learning (TL)-based models are fine-tuned concerning their hyperparameters using $\Psi_L(.,.)$, while the Deep Learning (DL)-based models undergo hyperparameter tuning through $\Psi_D(.,.,.)$. Following this, the function $\Psi_v(.,.)$ selects the top-performing eight models, denoted as B_m with $m = 1, 2, ..., 8$, from both the TL-based and DL-based models, based on the eval-

uation of the loss functions $Loss_z$, where z belongs to the set $\{L^{Fo}, L^{Cs}\}$. Here, L^{Fo} denotes the focal loss, and L^{Cs} represents the cross-entropy loss, as introduced in [140]. For addressing **RQ4**, the focal loss is employed to mitigate issues related to class imbalance, as outlined in Table 6.33.

In Figure 3.4, the transformer-based language models TL_j, where j belongs to the set {BanglaBERT, BELECTRA, $MuRIL$, $DeBERTa$, $MBERT$, XLM-RoBerta, IndicBERT, DistilBERT}, are evaluated alongside their corresponding initial hyperparameters $H_j^I \in \{$ *batch_size*, *epochs*, *max_sequence_length*$\}$ with respect to C_i. These models' hyperparameters are fine-tuned using $\Psi_L(.,.)$ through Eq. 3.2, yielding the outputs of TL-based tuned models as $TL_{ji}(H_j^T)$. Here, H_j^T represents the tuned hyperparameters for models j.

The DL-based models DL_k, where $k \in \{CNN, LSTM, BiLSTM, GRU\}$, along with their corresponding initial hyperparameters $H_k^I \in \{$ *batch_size*, *epochs*, *max_sequence_length*$\}$, utilize non-contextual embeddings $nED_l, l \in$ {GloVe, FastText, Word2Vec}, for corpus C_i to generate the DL-based tuned model $DL_{kli}(H_k^T)$, using Eq. 3.3 and the function $\Psi_D(.,.,.)$. Here, H_k^T represents the tuned hyperparameters for the DL-based model k.

The combination of eight TL-based models and five corpora resulted in 40 models. Additionally, combining five DL models, three embeddings, and five corpora produced 60 DL-based tuned models. These 100 models (i.e., 40 for TL and 60 for DL) are subject to the function $\Psi_v(.,.)$, which selects the best-performing eight models (B_m) based on their accuracy values using Eq. 5.21. *The best-performing models B_m are summarized in Table 6.33.* These selected models are then utilized for late fusion and statistical model ranking.

5.4.3.2 Late Fusion and Selection of the Best-performing Models

In this phase, two steps are involved. Firstly, the late fusion-based performance matrix is generated. Secondly, the best-performing four models are selected based on accuracy. The subsequent subsection provides a detailed description of these two steps.

Late Fusion This step employs the eight best-performing models (B_m) for late fusion. The combination choice $\binom{8}{3}$, i.e., the selection of 8 models taken three at a time. A total of 56 different combinations are formed for a single corpus and a single late fusion method using the Eq. 2.4 and function $\Phi_{pz}(.,)$. Each combination contains fusion-based statistics, encompassing accuracy, precision, recall, and average F1-score, and is defined as FM_{qz}. Considering all five corpora and two late fusion methods, namely *max-voting* and *averaging*, a total of 560

different combinations of FM_{qz} are generated, which is calculated as $\binom{8}{3} \times (i_s = 5 \times d_s = 2) = 560$. Here, i_s represents the size of the corpora set, and d_s denotes the total number of late fusion methods. Each combination includes 3 of the eight models, and their predictions are aggregated to create late fusion prediction. By leveraging these individual models' diversity and complementary strengths, late fusion aims to enhance the overall classification performance, providing more robust and accurate results. *The late fusion-based performance of five corpora is summarized in Table 6.34.* The 560 FM_{qz} and the 8 B_m models are then forwarded to the next step for model ranking based on the statistical significance test.

Best-performing Models' Selection In this step, we outline the model selection process for identifying the best-performing four models from the set of 560 late fusion-based performance matrices (FM_{qz}) and eight single performance matrices (B_m) using Eq. 2.5. The best four models, namely $RM_r, r = 1, ..., 4$, consistently demonstrate superior performance across *five corpora and performance metrics, shown in Table 6.34.* According to the performance summary in Table 6.34, the four best-performing models are $RM_{r(1)|r(1)=BanglaBERT}$, $RM_{r(2)|r(2)=MuRIL}$, $RM_{r(3)|r(3)=XLMRoBerta}$ and $RM_{r(4)|r(4)=DeBERTa}$. These best-performing four models will be further considered for attention-based early fusion, ensuring the selection of reliable and robust models to enhance the efficacy of our AFuNet text classification system.

5.4.3.3 Attention-based Early Fusion and Identification of Best-performing Models

This module takes the best-performing four models (RM_r) and the corpora C_i as input and produces an intermediate best-performing text classification model, denoted as $\Delta AFuNet$. The selection of $\Delta AFuNet$ involves two steps: first, the generation of early fusion models, and second, the choice of the $\Delta AFuNet$ using the statistical significance test (i.e., Welch's t-test and Friedman's test). The subsequent subsection provides a detailed description of each step.

Attention-based Early Fusion This study delves into the exploration of four advanced early fusion methods to address **RQ3**: concatenation (*cat*), averaging (*avg*), attention-based concatenation (*acat*), and attention-based averaging (*aavg*), across five corpora. The four best-performing models are generated by considering the combination choices of $\binom{4}{4}$, $\binom{4}{3}$ and $\binom{4}{2}$, yielding a total of 11 unique early fusion configurations for each method and corpus. The combination

choices of $\binom{4}{4}$, $\binom{4}{3}$, and $\binom{4}{2}$ entail selecting sets of models from a pool of four, incorporating all four models, groups of three, and pairs of two. This process comprehensively explores various model combinations, contributing to a thorough evaluation of their collective performance. In total, 220 models are generated, incorporating five corpora, four early fusion methods, and 11 combinations (i.e., $5 \times 4 \times (\binom{4}{4}+\binom{4}{3}+\binom{4}{2})$. The set of 220 early fusion-based models is then passed on to the subsequent section for selecting the best-performing intermediate model, denoted as $\Delta AFuNet$.

Best-performing Model Selection Selecting the best-performing model from the pool of 220 early fusion models involves subjecting them to two statistical significance tests: Welch's t-test and Friedman's. Welch's t-test serves to identify the most promising early fusion model by calculating t-values and p-values compared to other early fusion models using Eq. 5.22. This approach thoroughly assesses the performance of all models, ultimately selecting the optimal early fusion model through Welch's t-test. The results from Table 6.36 reveal that among the five corpora, the combination of $RM_{r(1)|r(1)=BanglaBERT}$ and $RM_{r(2)|r(2)=MuRIL}$ early fusion models outperforms others according to Welch's t-test. Similarly, the Friedman test, considering the performance of 220 early fusion models across multiple corpora (as shown in Table 6.35), also designates the pair of $RM_{r(1)|r(1)=BanglaBERT}$ and $RM_{r(2)|r(2)=MuRIL}$ early fusion models as statistically significant. The collective evidence from Tables 6.36 conclusively supports the notion that the early fusion of $RM_{r(1)|r(1)=BanglaBERT}$ and $RM_{r(2)|r(2)=MuRIL}$ models results in the development of the most influential text classification model, denoted as $\Delta AFuNet$. Subsequently, $\Delta AFuNet$ is advanced to the next section, which examines the impact of OOV words, class-imbalance issues, and ablation studies.

5.4.3.4 Ablation Study and Impact Analysis

The transition from $\Delta AFuNet$ to $AFuNet$ comprises two steps: ablation study and impact analysis. Initially, ablation studies involved adjustments of the dropout value (Dr), the number of attention layers (NAL), and the attention layer embedding dimension (AED). Subsequently, an impact analysis is undertaken to assess the effects of OOV words class-imbalanced corpora with monolingual and multilingual models on $\Delta AFuNet$. The following subsections elucidate these processes.

Ablation Studies The ablation studies comprehensively analyze the impact of various settings on the performance of $\Delta AFuNet$. This study considers three key factors: the dropout value (Dr), the number of attention layers (NAL), and the attention embedding dimensions (AED). By systematically altering these settings, we aimed to discern their individual and combined effects on the model's accuracy, precision, recall, and average F1-score using the function $\delta(.,.,.)$ in Eq. 5.23. This investigation sheds light on the optimal configurations for these settings, contributing to refining the proposed text classification model (i.e., $AFuNet$). Table 6.37 demonstrates the summary of the ablation studies.

Impact Analysis The impact analysis investigates the impact of three crucial factors on the $AFuNet$ performances. Firstly, it examines the influence of OOV words, which do not appear in the training vocabulary, on the model's accuracy, precision, recall, and average F1 score. Secondly, it delves into the consequences of class imbalance within the dataset, analyzing how differing class distributions affect the overall performance. Finally, we investigate the implications of employing monolingual versus multilingual language models. This assessment determines whether combining language models or focusing on domain-specific monolingual models can effectively elevate the fine-tuned capacity to classify text across various languages and domains. By conducting this comprehensive impact analysis, we gain insights into the enhanced capabilities of the optimized $AFuNet$ model, fine-tuned using the five corpora.

5.4.3.5 Architecture of AFuNet

AFuNet consists of three core processes: (i) Contextual feature extraction, (ii) Fine-tuning of attention-based early fusion, and (iii) Inference of attention-based early fusion. Figure 5.9 depicts a comprehensive overview of the AFuNet architecture.

Contextual Feature Extraction This phase is dedicated to transforming text into contextual features, which follows a sequential process comprising three key steps: (i) tokenization, (ii) extraction of token to contextual features, and (iii) element-wise addition of contextual features and position encoding.

In the first step, the training set for corpus C_i is denoted as $C_i^{tr} = \{X_{i1}^{tr}, X_{i2}^{tr}, ..., X_{ig}^{tr}\}$, where $g = 1, ..., G^{tr}$, and the testing set is represented as $C_i^{ts} = \{X_{i1}^{ts}, X_{i2}^{ts}, ..., X_{ip}^{ts}\}$, where $p = 1, ..., G^{ts}$. Here, X_{ig}^{tr} refers to the g^{th} training sample of corpus i, and X_{ip}^{ts} refers to the p^{th} testing sample of corpus i. The values G^{tr} and G^{ts} denote the total number of training and testing samples in corpus i.

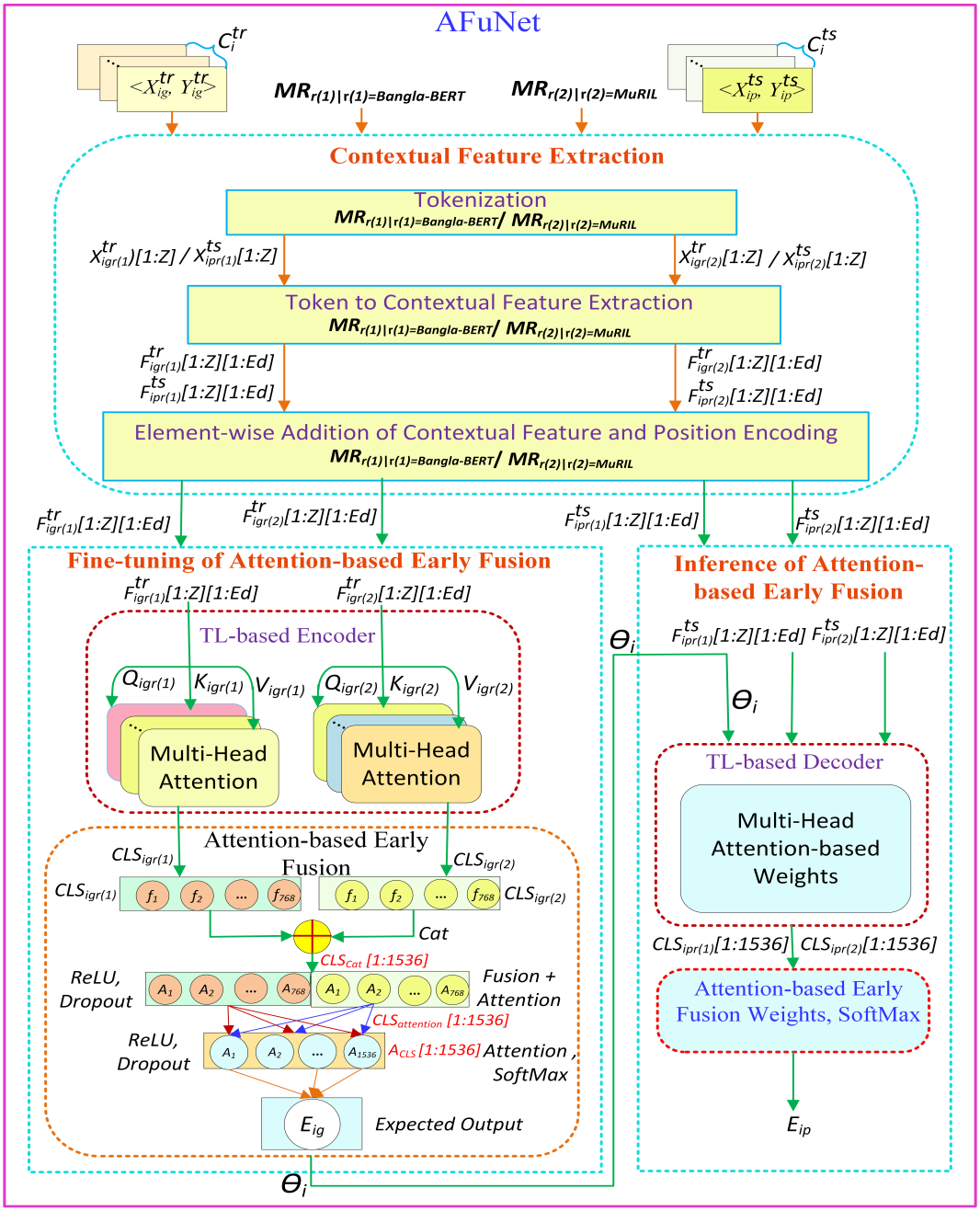


Figure 5.9: Architecture of AFuNet: Text classifier for a low-resource language

The second step involves tokenizing the g^{th} training sample, X_{ig}^{tr}, using the models $MR_{r(1)|r(1)=BanglaBERT}$ and $MR_{r(2)|r(2)=MuRIL}$. The tokenized tokens based on $MR_{r(1)|r(1)=BanglaBERT}$ are represented as $X_{igr(1)}^{tr}[1:Z] =< T_{ig1}^{tr}, T_{ig2}^{tr}, ..., T_{igZ}^{tr} >$, while those based on $MRr(2)|r(2) = MuRIL$ are denoted as $X_{igr(2)}^{tr}[1:Z] =< T_{ig1}^{tr}, T_{ig2}^{tr}, ..., T_{igZ}^{tr} >$. Here, Z signifies the total number of tokens in the g^{th} sample. Similar tokenized vectors are obtained for the p^{th} testing sample, Xip^{ts}, and the models $MR_{r(1)|r(1)=BanglaBERT}$ and $MR_{r(2)|r(2)=MuRIL}$. Moving to the second step, the tokenized vectors for training samples ($X_{igr(1)}^{tr}[1:Z]$ and $X_{igr(2)}^{tr}[1:Z]$)

and testing samples ($X^{tr}_{ipr(1)}[1:Z]$ and $X^{tr}_{ipr(2)}[1:Z]$) are subjected to token to contextual feature extraction, resulting in train sample feature matrices ($F^{tr}_{igr(1)}[1:Z][1:E_d]$ and $F^{tr}_{igr(2)}[1:Z][1:E_d]$) and test sample feature matrices ($F^{ts}_{ipr(1)}[1:Z][1:E_d]$ and $F^{ts}_{ipr(2)}[1:Z][1:E_d]$). These matrices are in $\mathbb{R}^{(Z\times E_d)}$, where Z indicates the total number of tokens, and E_d represents the contextual embedding dimension.

In the third step, the feature matrices are further processed by adding contextual features and position encoding element-wise. Specifically, the encoding for even position is represented as $sin(\frac{Z}{10000^{(2\times Z/E_d)}})$ and odd position is represented as $cos(\frac{Z}{10000^{(2\times Z+1/E_d)}})$. For instance, the first even position element-wise addition of the contextual feature and position encoding for a training sample is expressed as $F^{tr}_{igr(1)}[2:2][1:E_d] = sin(\frac{Z=2}{10000^{(2\times Z=2/E_d)}}) + (F^{tr}_{igr(1)}[2:2][1:E_d])$. Similarly, the first odd position element-wise addition of training sample contextual feature and position encoding is expressed as $F^{tr}_{igr(1)}[1:1][1:E_d] = cos(\frac{Z=1}{10000^{(2\times Z=1+1/E_d)}}) + (F^{tr}_{igr(1)}[1:1][1:E_d])$. After the element-wise addition, these feature matrices are forwarded to the fine-tuning (i.e., $F^{tr}_{igr(1)}[1:Z][1:E_d]$ and $F^{tr}_{igr(2)}[1:Z][1:E_d]$) and inference (i.e., $F^{ts}_{ipr(1)}[1:Z][1:E_d]$ and $F^{ts}_{ipr(2)}[1:Z][1:E_d]$) phases.

Fine-tuning of Attention-based Early Fusion Figure 5.9 illustrates the fine-tuning of the two best-performing transformer-based language models, namely $MR_{r(1)|r(1)=BanglaBERT}$ and $MR_{r(2)|r(2)=MuRIL}$, using the training set C^{tr}_i to generate the attention-based early fusion model Θ_i. The contextual feature matrices $F^{tr}_{igr(1)}[1:Z][1:E_d]$ and $F^{tr}_{igr(2)}[1:Z][1:E_d]$ of the g^{th} training sample $< X^{tr}_{ig}, Y^{tr}_{ig} > \in C^{tr}_i$ are input to the *TL-based Encoder* block. Within this encoder block, the multi-head attention module calculates self-attention using the [141] methodology, with each head conducting its self-attention computation. The resulting self-attention matrices are then combined with the multi-head attention module. These matrices proceed through the *add and normalization* module, followed by the *feed-forward module*, culminating in the generation of global and local holistic classification embedding vectors, denoted as $CLS_{igr(1)} \in \mathbb{R}^{(E_d)}$ and $CLS_{igr(2)} \in \mathbb{R}^{(E_d)}$, respectively. In this context, E_d represents the embedding vector dimension.

In the **attention-based early fusion block**, the input consists of two embedding vectors: $CLS_{igr(1)} = [f_1, f_2, ..., f_{768}]$ and $CLS_{igr(2)} = [f_1, f_2, ..., f_{768}]$, yielding the expected class name E_{ig}. These two vectors are fused using the concatenation operation (**cat(.)**) via Eq. 5.26:

$$CLS_{cat}[1:1536] = CLS_{igr(1)} \oplus CLS_{igr(2)} \tag{5.26}$$

Here, $CLS_{cat}[1:1536]$ represents the resulting fused embedding feature vector. This fused vector is then passed to the subsequent layer for the calculation of a single-head self-attention mechanism using Eq. 5.27:

$$CLS_{attention}[1:1536] = softmax\left(\frac{Q_{CLS} \times K_{CLS}}{\sqrt{E_d}}\right) \cdot CLS_{cat}[1:1536] \quad (5.27)$$

Here, $CLS_{attention}[1:1536]$ denotes the attention-based fused vector. Where, $Q_{CLS} \in \mathbb{R}^{(E_d \times E_d)}$ and $K_{CLS} \in \mathbb{R}^{(E_d \times E_d)}$ represent the query and key matrices, respectively, and $E_d = 768$ is the embedding dimension. The attention operation is succeeded by a ReLU activation and dropout operation, effectively mitigating overfitting concerns and enhancing the generalization of model performance. The resultant vector, denoted as $CLS_{attention}[1:1536]$, undergoes successive attention and softmax operations by Eqs. 5.28-5.29.

$$A_{CLS}[1:1536] = \text{softmax}\left(\frac{Q_{CLS1} \times K_{CLS1}}{\sqrt{E_d}}\right) \cdot CLS_{attention}[1:1536] \quad (5.28)$$

$$E_{ig} = \text{softmax}\left(A_{CLS}[1:1536]\right) \quad (5.29)$$

In this context, the variables within the equations, $A_{CLS}[1:1536]$ and E_{ig}, represent the attention-based normalized vector and the expected output of the g^{th} sample of corpus i, respectively. The subsequent backpropagation operation adjusts the trainable variable weights based on the ground truth value Y_{ig}^{tr} and the expected value E_{ig}. This adjustment process generates the best-performing model Θ_i, which is then utilized for inference within the attention-based early fusion module.

Inference of Attention-based Early Fusion The performance assessment of the attention-based early fusion model occurs when using the test set C_i^{ts}. The best-performing model Θ_i and feature matrices $X_{ipr(1)}[1:Z][1:E_d]$ and $X_{ipr(2)}[1:Z][1:E_d]$ corresponding to the test sample X_{ip}^{ts} generats an expected text category name E_{ip} specific to corpus i and sample p. The TL-based decoder module securely retains the finely tuned weight matrices. It operates by independently projecting the two input feature matrices using the attention-based weights generated from the generalized embedding feature vectors $CLS_{ipr(1)}$ and $CLS_{ipr(2)}$, respectively. These two embedding vectors are then combined via the attention-based early fusion weights. Subsequently, the Softmax function is applied to generate the expected text category name E_{ip}. This process is employed across the test sample set C_i^{ts}, which is represented as $\{X_{i1}^{ts}, X_{i2}^{ts}, ..., X_{ip}^{ts}\}$ with $p = 1, ..., G^{ts}$. This operation yields an expected category set $\{E_{i1}, E_{i2}, ..., E_{ip}\}$.

The subsequent statistical performance summary is generated by comparing this expected category set with the ground truth set, denoted as $\{Y_{i1}^{ts}, Y_{i2}^{ts}, ..., Y_{ip}^{ts}\}$. The following subsections provide an in-depth discussion of the performance of AFuNet for each corpus C_i. *The comprehensive performance of the AFuNeT model and a comparison of its performance with other models will be discussed in Section* 6.5.5.2.

While the proposed AFuNeT achieved state-of-the-art accuracy for five Bengali text classification corpora (BARD, OSBC, BTCC1, Prothom-Alo, and Indic-NLP), for a more comprehensive assessment, we have also applied the AFuNeT architecture, specifically the attention-based early fusion model, to four additional Bengali authorship attribution tasks. The subsequent subsection will elaborate on the use of attention-based early fusion for Bengali authorship attribution.

5.4.3.6 Attention-based Early Fusion for Bengali Authorship Attribution

The main goal of this research is to develop an *intelligent* authorship attribution system (i.e., AuthorNet) that can intelligently classify a piece of Bengali author text based on a predefined author name. The AuthorNet comprises four main modules, i.e., (i) Automatic Hyperparameters Tuning, (ii) Best-performing Model Selection, (iii) Training Attention-based Early Fusion Model, (iv) Inferencing by Attention-based Early Fusion Model. Figure 5.10 shows the abstract view of the AuthorNet system. Each of the modules is described in the following subsections.

Automatic Hyperparameters Tuning In recent years, the language model achieved state-of-the-art results for many NLP downstream tasks [16, 18]. Still, the performance of fine-tuning pre-trained language strongly depends on hyperparameters [142]. This is the philosophy of the proposed study, which carried out the pre-trained language model fine-tuning impact on low resource authorship attribution task (e.g., Bengali AA). Due to the hardware limitation (e.g., GPUs) and optimization time budget, this study investigates the three most influential hyperparameters' impacts on low-resource pre-trained language model fine-tuning tasks. The hyperparameters are epoch number, batch size, and maximum sequence length. To answer the **RQ3**, we developed an automatic hyperparameters tuning Algorithm. The inside of the Algorithm is illustrated in Algorithm Algorithm 10.

From Algorithm 10, hyperparameters tuning function $\lambda(.,)$ takes two inputs: first input is hyperparameters of transformer-based language model $TL_l(H_l^I)$ and

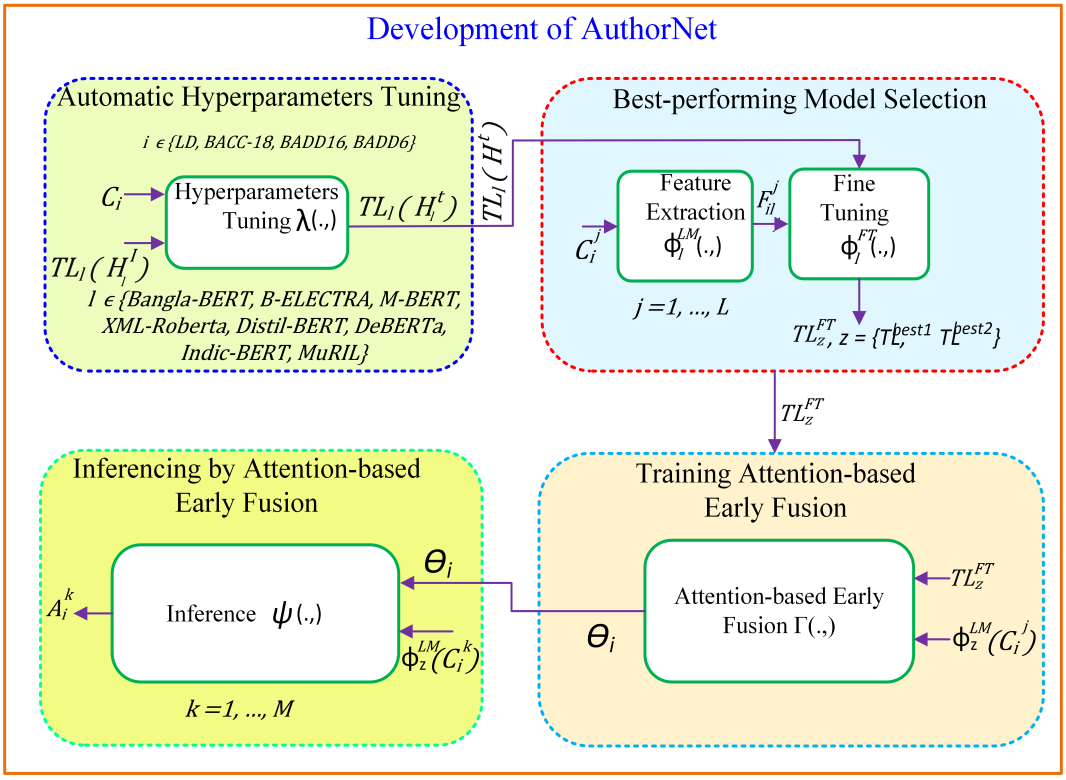


Figure 5.10: Abstract view of the development of AuthorNet

the second one is corpus C_i. The $TL_l, l \in$ {M-BERT, Bengali-BERT, M-Distil-BERT, Bengali-ELECTRA, Indic-BERT, MuRIL, XML-Roberta, AraBERT} denotes the transformer-based language models and H_l^I denotes the initial hyperparameters set. Whereas $C_i, i \in \{TextClassificationCorpora\}$ indicates the corpus set. The output of this module is the tuned hyperparameters, i.e., $TL_l(H_l^t)$. These tuned hyperparameters are forwarded to the authorship attribution training module.

Best-performing Model Selection: The main goal of this module is to generate the authorship attribution models using the tuned hyperparameters and select the best-performing models for attention-based early fusion training and inferencing. This module comprises two components: (i) Feature extraction and (ii) Fine-tuning. The following subsections describe each of the components.

Feature Extraction: To address **RQ2**, this study leverages transformer-based language models to extract text features and effectively overcome the challenges associated with out-of-vocabulary (OOV) words. During the text-to-feature extraction phase, the input comprises a collection of author texts, while the output consists of sets of 2D feature matrices. The j^{th} training sample of i^{th} corpus C_i^j is produced the feature matrix F_{il}^j, $F_{il}^j \in \mathbb{R}^{(sl \times D_f)}$ using l^{th} transformer-based

language model. Where sl represents the maximum sequence length and D_f represents the feature dimension of j^{th} test sample. Fig. 5.11 shows the details of the author's text to feature extraction. Each of the text is initially converted

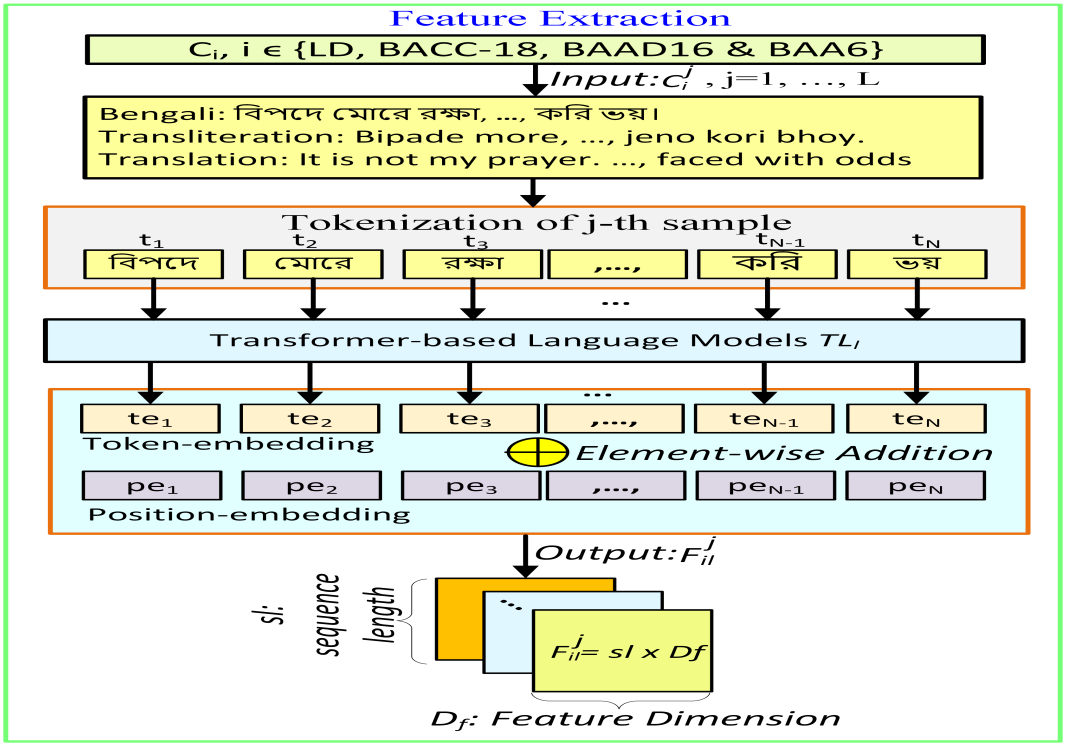


Figure 5.11: Overview of text-to-feature extraction

to a set of tokens, i.e., $\{t_1, t_2, ..., t_N\}$, where t_N denotes the N^{th} token of the given input text C_i^j. This set of tokens is passed through the transformer-based language model, and each token is converted to a numerical feature vector, i.e., token embedding using Eq. 5.30.

$$\mathcal{F}_{il}^j = \Phi_l^{LM}(C_i^j), j = 1, ..., L; i \in \{LD, BACC-18, BAAD16, BAAD6\} \quad (5.30)$$

C_i^j represents the j^{th} sample for corpus i and corresponding feature matrix is $F_{il}^j \in \mathbb{R}^{sl \times D_f}$. The sl and D_f denote the maximum sequence length and corresponding dimension for features. The TL_{il}^{FT} represents the attribution model for corpus i. Among these models, the two best-performing models $TL_z^{FT}, z = \{TL^{best1}, TL^{best2}\}$ are selected for attention-based early training and inferencing.

The $\{te_1, te_2, ..., te_N\}$ indicates the token embedding set and $te_N \in \mathbb{R}^{D_f}$ represent the N^{th} token embedding of D_f feature dimension. To improve the word-word correlation, the position embedding set $\{pe_1, pe_2, ..., pe_N\}$ is element-wise added with the token embedding set, i.e., $\{te_1, te_2, ..., te_N\} \oplus \{pe_1, pe_2, ..., pe_N\}$

where $\oplus$ indicates the element-wise addition. Finally, the input text C_i^j is converted to a 2D feature matrix, i.e., F_{il}^j, which is used for the fine-tuning phase.

Fine-tuning: The fine-tuning module takes a set of feature matrices $[F_{il}^j]_{j=1}^{j=L}$ as input and generates two best-performing attribution models namely $TL_z^{FT}, z = \{TL^{best1}, TL^{best2}\}$. Here, i represents the corpus i, z indicates the best-performing language model z and L denotes the total number of training samples in the corpus.

The input feature matrix is passed through multiple attention heads. Each head calculates three attention vectors: query ($Q_{il} \in \mathbb{R}^{D_f}$), key ($K_{il} \in \mathbb{R}^{D_f}$), and value ($V_{il} \in \mathbb{R}^{D_f}$) using trainable attention matrices: query weight matrix $WQ_{il} \in \mathbb{R}^{(D_f \times D_f)}$, key weight matrix $WK_{il} \in \mathbb{R}^{(D_f \times D_f)}$, and value weight matrix $WV_{il} \in \mathbb{R}^{(D_f \times D_f)}$, respectively. These attention vectors, along with the feature matrix F_{il}^j, are passed through layer normalization and feed-forward network blocks, ultimately producing the classifiable feature vector $CLS \in \mathbb{R}^{D_f}$.

The classifiable feature vector is then fed into a Multilayer Perceptron (MLP) block, which generates the output corresponding to the expected author name from a predefined set of authors. The error value is calculated based on the output and used to update the trainable weight matrices through backpropagation. Consequently, the corpus i produces the authorship attribution model TL_{il}^{FT} using the transformer-based language model l. Among the eight fine-tuned attribution models, the attention-based early fusion training and inferencing module selects the best-performing model $TL_z^{FT}, z = \{TL^{best1}, TL^{best2}\}$. This study, based on the accuracy and statistical significance tests (i.e., Details in Sec. 6.5.5.3), the first best-performing model is MuRIL (i.e., TL^{best1}=MuRIL) and second best-performing model is Bangla-BERT (i.e., TL^{best2}=Bangla-BERT). This selection process is depicted in Figure 5.10, and the results are summarized in Table 6.42. These two models, TL^{best1} = (MuRIL) and TL^{best2} = (Bangla-BERT), are subsequently utilized in the attention-based early fusion training phase.

Training Attention-based Early Fusion: To address **RQ3**, this module introduces an attention-based early fusion approach that combines the two best-performing models $TL_z^{FT}, z = \{TL^{best1}, TL^{best2}\}$. The feature matrices of these models are represented as F_{iz}^j, and j represents the j^{th} test sample from the corpus i. The output of this module is an attention-based early fusion authorship attribution model, referred to as Θ_i. The attention-based early fusion training and inferencing module for authorship attribution is visually depicted in Figure 5.12.

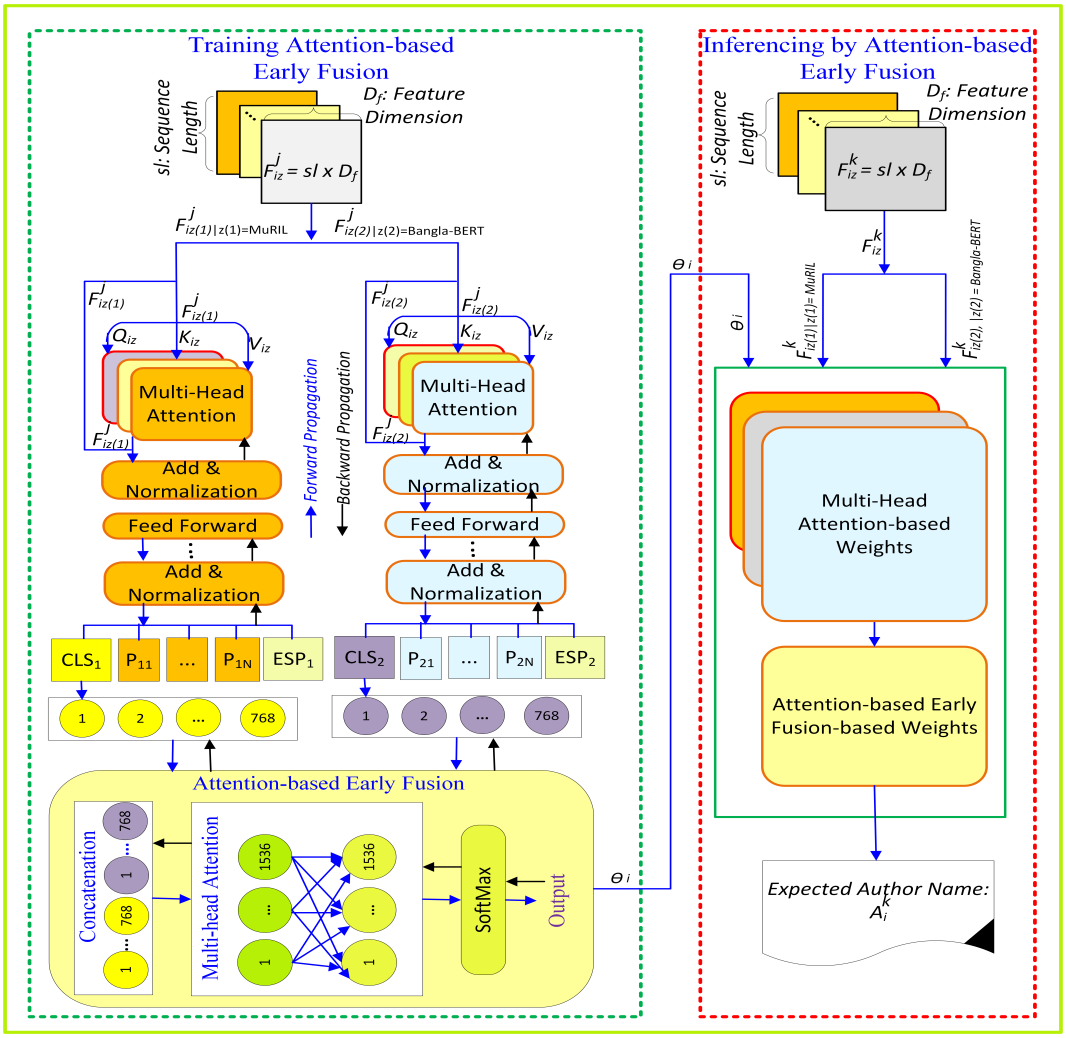


Figure 5.12: Intelligent AuthorNet: Overview of attention-based early fusion of authorship attribution training and inferencing

In Figure 5.12, the j^{th} training sample of the corpus i is represented by the MuRIL-based feature matrix denoted as $F^{j}_{iz(1)|z(1)=MuRIL}$, and the Bangla-BERT based feature matrix, denoted as $F^{j}_{iz(2)|z(2)=Bangla-BERT}$. The MuRIL-based feature encoding is represented by CLS_1, P_{11}, ..., P_{1N}, and ESP_1, while the Bangla-BERT based feature encoding is represented by CLS_2, P_{21}, ..., P_{2N}, and ESP_2. Here, CLS_1 and CLS_2 refer to single-dimensional vectors. Similarly, P_{1N} and P_{2N} represent the N^{th} token feature vectors of MuRIL and Bangla-BERT models, respectively, where (P_{1N}, P_{2N}). The CLS_1 and CLS_2 serve as document-level stylometric feature representations for the j^{th} sample and are forwarded to the attention-based early fusion module. These two document-level feature vectors are fused using the concatenation operation, denoted as $CLS_1 \oplus CLS_2$. The concatenated feature vector is then passed through the multi-head attention block.

The attention block calculates the attention value using Eq. 5.31.

$$A_{iz}^{V} = (\frac{CLS_{iz}^{Q} \times CLS_{iz}^{K}}{\sqrt{dim(CLS_z)}}) \times CLS_z \tag{5.31}$$

In this context, A_{iz}^{V} represents the attention-based vector, where the length of A_{iz}^{V} is *1536*. CLS_{iz}^{Q} and CLS_{iz}^{K} refer to the model z's trainable query and key vectors, respectively. CLS_z denotes the encoded vector of the model z, and the dimension of CLS_z is denoted as $dim(CLS_z)$. The A_{iz}^{V} vector is normalized using the ReLU function and passed to the subsequent fully connected layer. The attention-based early fused vector is then propagated through the $SoftMax$ layer to generate author-wise expected values. Deviations or errors from these values are calculated, and the model weights are adjusted through the backward pass operation. Finally, the fusion-based authorship attribution model Θ_i is forwarded to the inference of the Attention-based early fusion module.

Inferencing by Attention-based Early Fusion: This module takes two inputs, i.e., a set of test samples $[C_i^k]_{k=1}^{k=M}$ and an attention-based early fusion model, Θ_i and produces one output, i.e., expected author name A_i^k. Here M denotes the total number of test samples in the corpus i, and z indicates the transformer-based model z. The k^{th} test feature matrix F_{iz}^k is produced by the function $\Phi_z^{LM}(.,)$ for the test sample C_i^k and fed to the attribution model Θ_i that generates an expected author list A_i^k. This author list is forwarded to the $softMax$ block, which generates the expected author name A_i^k from the predefined author list based on the maximum expected value. In this way, the corpus i measures the performances of the attention-based early fusion fine-tuned transformer-based attribution model. *The comprehensive performance of the AuthorNet model and a comparison of its performance with other models will be discussed in Section* 6.5.5.3.

5.4.3.7 Multimodal Text Classification

This research explores the influence of multimodal input on low-resource text classification, specifically in Bengali. Figure 5.13 illustrates the abstract view of the multimodal text classification system (MMTC). The MMTC consists of three primary modules: (i) BanglaBERT-based text feature extraction, (ii) ViT-based image feature extraction, and (iii) Attention-based early fusion. Each module is described in detail in the following sections.

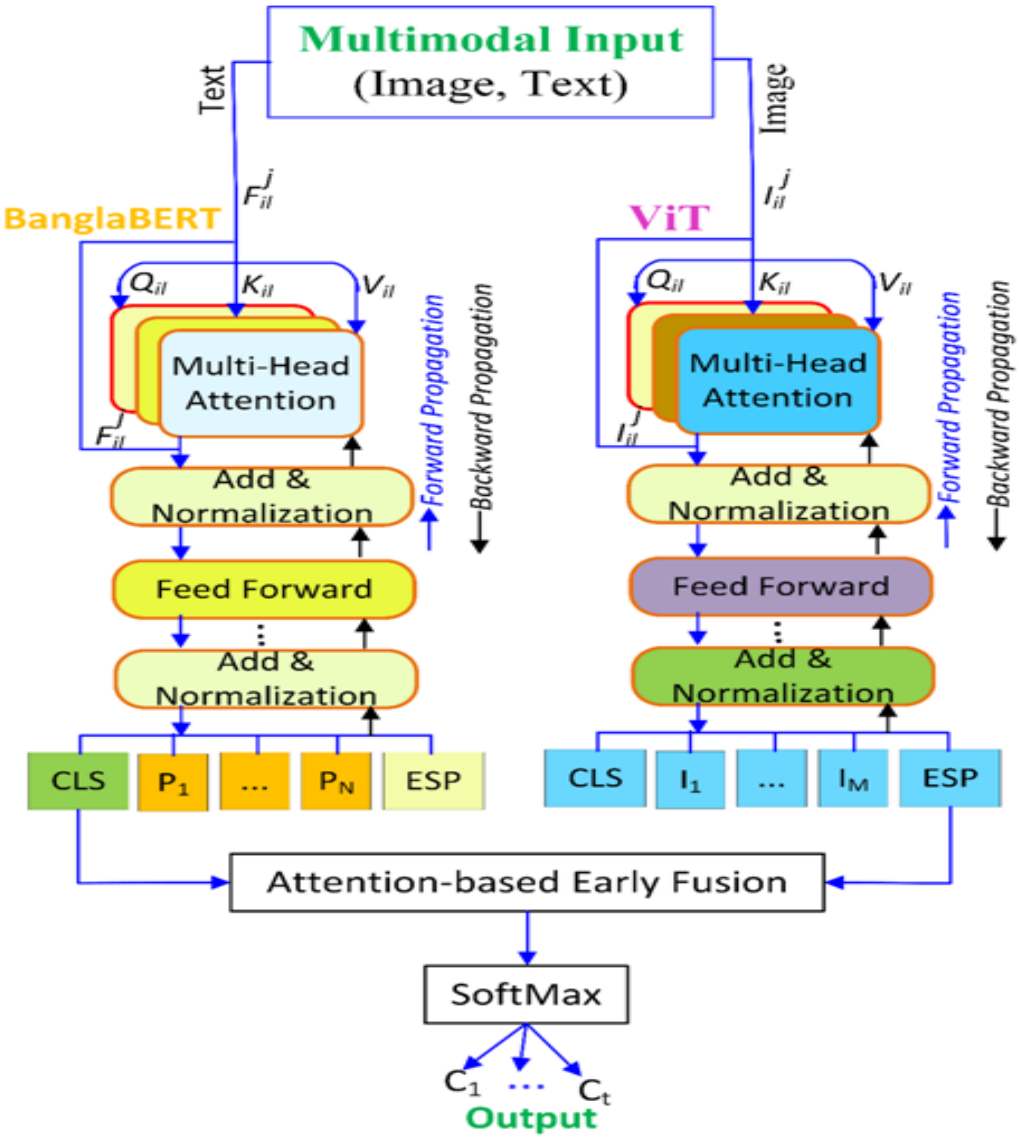


Figure 5.13: Abstract view of multimodal text classification system

(i) BanglaBERT-based text feature extraction In this module, the i^{th} corpus of the j^{th} sample F_{il}^{j} is inputted into the BanglaBERT model, which generates features using Equation 5.32.

$$ZF_{il}^{j} = BanglaBERT(F_{il}^{j}) \tag{5.32}$$

The ZF_{il}^{j} represents the encoded feature matrix, where $ZF_{il}^{j} =< CLS, P_1, ..., P_N, ESP >$. Here, CLS denotes the classifiable text feature, and P_N represents the feature of the N^{th} word. The CLS feature comprises a 768-dimensional feature vector, which is then passed to the attention-based early fusion module.

(ii) ViT-based image feature extraction In this module, the i^{th} corpus of the j^{th} sample image I_{il}^{j} is inputted into the ViT model and extracted the classifiable visual features using Eq. 5.33

$$IF_{il}^{j} = ViT(I_{il}^{j}) \tag{5.33}$$

The IF_{il}^{j} represented the encoded feature matrix and $IF_{il}^{j} =< CLS, I_1, ..., I_M, ESP >$. Where CLS represented the classifiable image feature vector and I^M indicated the M^{th} patch feature vector. The CLS features forwarded to the attention-based early fusion module.

(iii) Attention-based early fusion In the fusion module, BanglaBERT-based text features and ViT-based image features are integrated using attention-based early fusion techniques. For BanglaBERT-based text feature extraction, the i^{th} corpus of the j^{th} sample $F^{j}il$ is fed into the BanglaBERT model, generating encoded feature matrix $ZFil^{j}$. This matrix comprises key elements such as CLS, representing the classifiable text feature, and P_N denoting the feature of the N^{th} word. The CLS feature, a 768-dimensional vector, is then directed to the attention-based early fusion module. Similarly, for ViT-based image feature extraction, the i^{th} corpus of the j^{th} sample image $I^{j}il$ is inputted into the ViT model to extract classifiable visual features, represented by encoded feature matrix $IFil^{j}$. This matrix includes CLS, indicating the classifiable image feature vector, and I^M representing the feature of the M^{th} patch. The CLS features from both modalities are then fused in the attention-based early fusion module. The fused features are then fed to the SoftMax layer and produced the expected value of each category, i.e., $< C_1, ..., C_t >$ where t indicated the total number of category in the multimodal text classification corpus. The maximum expected value indicated the deserted category name. The performance of Bengali multimodal text classification is presented in Section 6.5.9.

5.5 Summary

In this study, we employed three distinct text classification techniques Statistical, Deep Learning, and Transformer-based fusion to facilitate the development of text classification models. Table 5.1 provides a comprehensive summary of the tasks, model names, and corresponding corpora.

From Table 5.1, the BDTCC corpus was utilized for document classification, and classification models were generated using Statistical (ST), Deep Learning

Table 5.1: Text classification models development summary, ST: Statistical model, DL: Deep Learning-based Model, FN: Attention-based Early Fusion Networks, Y: Yes/Developed Model, N: Not/Not Developed Model

Corpora	Tasks	Techniques			
		ST	DL	TL	FN
BDTCC	Document Classification	Y	Y	Y	N
BACC-18, BAAD16, BAAD16, LD	Authorship Attribution	Y	Y	Y	Y
BTCC11, BARD, IndicNLP, Prothom-ALo, OSBC	Document Classification	Y	Y	Y	Y
BCoVC	Bengali Covid Text Classification	Y	Y	Y	N
AraCoV	Arabic Covid Text Classification	N	Y	Y	N
ECoVC	English Covid Text Classification	Y	Y	N	N

(DL), and Transformer-based Language Model (TL) techniques. For authorship attribution tasks, the BACC-18, BAAD16, BAAD16, and LD corpora were employed, leading to the generation of attribution models through Statistical (ST), Deep Learning (DL), Transformer-based Language Models (TL), and attention-based early fusion Networks (FN). Meanwhile, the BTCC11, BARD, IndicNLP, Prothom-Alo, and OSBC corpora were used for Bengali document classification tasks, involving all techniques (ST, DL, TL, and FN). Specific Covid text classification corpora, namely BCoVC and AraCoV, were used for Covid text identification in Bengali and Arabic languages, generating identification models through ST, DL, and TL techniques. English Covid text classification models were created using TL and DL techniques with the ECoVC corpus. The performances of these techniques and models are presented in Section 6.5 and Section 6.5.5.

Chapter 6

Experimental Results and Discussion

The deployment environment and evaluation measures for assessing a resource-constrained text classification system are of paramount importance. This chapter initiates by providing an overview of the experimental setup, emphasizing the deployment environment specific to the resource-constrained text classification system. Following this, we elaborate on the performance metrics employed to assess the system's effectiveness. Lastly, we present the results encompassing both baseline and proposed models, along with an in-depth analysis of errors within the proposed system and a discussion of its limitations.

6.1 Experiment Settings and Environments

This research aimed to develop a system called 'AFuNeT' for resource-constrained text classification tasks, including Bengali, utilizing various language models implemented through Python-based packages within the PyTorch framework. A comparative analysis was conducted, pitting the proposed model against several baseline methods, encompassing Convolutional (CNN, ACNN, and VDCNN), Sequential (LSTM BiLSTM, and GRU), and Transformer-based language models, such as Bangla-BERT, B-ELECTRA, M-BERT, XLM-RoBerTa, DistilBERT, DeBERTa, Indic-BERT, and MuRIL. The implementations of both the proposed model and the baseline methods leveraged PyTorch 1.9.0, Pandas, Sklearn libraries, Python3 (version: 3.6), Numpy, and Transformer (version: 4.9.0). In terms of hardware, the study employed a multi-core processor (core-i7) coupled with an NVIDIA RTX 3050 GPU featuring 8GB of internal GPU memory, alongside 32GB of physical memory.

6.2 Evaluation Measures

To assess the performance both the proposed and baseline models, we employed a combination of intrinsic and extrinsic evaluation measures. The subsequent paragraphs describe each of these measures.

6.2.1 Intrinsic Evaluators

The intrinsic evaluators evaluate each word embedding model's word-level semantic, syntactic, relatedness, or analogy task performance. This evaluation helps to decide the best-suited embedding model for the downstream task that requires minimum time and memory usage. The semantic (C_{S_s}), syntactic (C_{S_y}) and relatedness (C_{S_r}) similarity measure is calculated using Eq. 6.1.

$$C_{cs}(A_w, B_w) = \frac{\overrightarrow{A_w} \cdot \overrightarrow{B_w}}{||\overrightarrow{A_w}|| \times ||\overrightarrow{B_w}||}, cs = [S_s, S_y, S_r] \tag{6.1}$$

Here, A_w and B_w denote the semantic, syntactic, or relatedness first and second words of the intrinsic datasets, respectively. The feature vector of word A_w and B_w represented by $\overrightarrow{A_w}$ and $\overrightarrow{B_w}$ respectively. C_{cs} presents the Cosine similarity score of $cs \in \{S_s, S_y, S_r\}$. The average Cosine similarity score of semantic, syntactic, and relatedness datasets are calculated using Cosine similarity score C_{cs}, which are represented by $\overline{C}_{S_s}$, $\overline{C}_{S_y}$ and $\overline{C}_{S_r}$ respectively. In this study, we also measure the Spearman ($\hat{\rho}$) and Pearson ($\hat{r}$) correlations [143] using the Cosine similarity and human judgment scores.

The word analogy also measures the embedding model performance using the pair-wise word alikeness, such as: if word A_w is to be word B_w and word C_w is to be word D_w then pair (A_w:B_w) is alike (C_w:D_w). The word alikeness problem is solved by the 3COSADD [144], and 3COSMULL arithmetic formulations [145]. For this purpose, given this (A_w:B_w :: C_w : $-$) then find the best match word for the blank $-$ (i.e., D_w) such that (A_w : B_w) is alike (C_w : D_w). To solve this problem, the 3COSADD finds the best matching word D_w using Eq. 6.2.

$$D_w = \underset{D_w \in V}{max}(C_{cs}(D_w, C_w) - C_{cs}(D_w, A_w) + C_{cs}(D_w, B_w)), cs = [A_t] \tag{6.2}$$

Here V is the total number of vocabularies in the embedding model. Another variation of this solution is 3COSMULL to find the best-matching word D_w using Eq. 6.3.

$$D_w = \underset{D_w \in V}{max} \frac{C_{cs}(D_w, C_w) \times C_{cs}(D_w, B_w)}{C_{cs}(D_w, A_a) + \epsilon}, cs = [A_t] \tag{6.3}$$

Here, ϵ is a small (i.e., 0.000001) value used for overcoming the division by zero. For calculating the arithmetic correlation of D_w with other three words, Eq. 6.2 or 6.3 is used, whereas Eq. 6.1 is used to compute Cosine similarity. The word analogy task performance is calculated by the ratio of $\frac{Acc}{len(A_t)}$, where Acc indicates the total number of deserted words D_w found and $len(A_t)$ represents the length of the analogy task.

6.2.2 Extrinsic Evaluators

The extrinsic evaluators assess the text classification task performance of the models. The accuracy and error of the baselines and proposed models are estimated by several statistical metrics such as accuracy (A), precision (P), and recall (R), along with the weighted averages of precision (WA_p) and recall (WA_r). Additionally, we scrutinized macro-average precision (M_{ap}) and macro-average recall (M_{ar}) metrics, as well as the F1-score (F1), which collectively provide a holistic view of performance. Furthermore, to ascertain the statistical significance and robustness of our findings, we incorporated the p and t values of Welch's t-test (Wp and Wt) and Friedman's test (Fp and Ft) [146]. Moreover, we conducted ablation analyses across the five corpora to unravel the intricacies of model performance. This comprehensive evaluation framework ensures a thorough and reliable assessment of the proposed model against established benchmarks and methodology.

Eqs. 6.4 - 6.7 are used to calculate the values of p, r, F_1 and a respectively.

$$\text{Accuracy (A)} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Predictions}} \tag{6.4}$$

$$\text{Precision (P)} = \frac{\text{True-Positives}_i}{\text{True-Positives}_i + \text{False-Positives}_i} \tag{6.5}$$

$$\text{Recall (R)} = \frac{\text{True-Positives}_i}{\text{True-Positives}_i + \text{False-Negatives}_i} \tag{6.6}$$

$$WeightedAveragePrecision(WA_p) = \frac{\sum_{i=1}^{n} w_i \cdot P_i}{\sum_{i=1}^{n} w_i} \tag{6.7}$$

$$WeightedAverageRecall(WA_r) = \frac{\sum_{i=1}^{n} w_i \cdot R_i}{\sum_{i=1}^{n} w_i} \tag{6.8}$$

$$Macro - AveragePrecision(M_{ap}) = \frac{1}{n}\sum_{i=1}^{n} P_i \tag{6.9}$$

$$Macro - AverageRecall(M_{ar}) = \frac{1}{n}\sum_{i=1}^{n} R_i \tag{6.10}$$

In these equations, n represents the number of classes or categories, w_i is the weight associated with each class, True-Positives$_i$ is the number of true positive predictions for class i, False-Positives$_i$ is the number of false positive predictions for class i, and False-Negatives$_i$ is the number of false negative predictions for class i.

6.3 Results

The performance evaluation of the developed resource-constrained text classification system involves three primary aspects: (i) analysis of corpora development, (ii) performance of the embedding models, and (iii) performance of the text classification models. Detailed analysis of corpora development has already been presented in Chapter 3. The remaining aspects are discussed in the following sections.

6.4 Performance of Embedding Models: Intrinsic Evaluation

In this research, we have developed three types of embedding models: (i) Individual embedding, (ii) Meta-Embedding, and (iii) Attention-based Meta-Embedding. The following subsections describe each of the embedding performances.

6.4.1 Performance of Individual Embeddings

There are three embedding corpora, i.e., Bengali embedding corpus (BEC), Authorship word embedding corpus (WEC), and Bengali Covid embedding corpus (CovEC) are generated the embedding models using three individual embedding methods such as GloVe, FastText, and Word2Vec. In the following, each of the individual embedding performances is described.

(i) BEC-based Individual Embeddings Performances Nineteen individual embedding models (6 for Word2Vec, 7 for FastText & 6 for GloVe) have been evaluated using these statistical measures. Table 6.1 shows the results of the four best-performing individual embeddings.

Table 6.1: Spearman ($\hat{\rho}$), Pearson ($\hat{r}$) and average cosine similarity (A_{CS}) statistics for 200 semantic, syntactic & relatedness word pairs.

Models	Semantic(%)			Syntactic(%)			Relatedness(%)		
	$S_{s\hat{\rho}}$	$S_{s\hat{r}}$	A_{CS}	$S_{y\hat{\rho}}$	$S_{y\hat{r}}$	A_{CS}	$S_{r\hat{\rho}}$	$S_{r\hat{r}}$	A_{CS}
GloVe	**64.47**	**65.29**	**76.25**	**69.10**	**74.97**	**78.53**	**79.90**	**80.14**	**85.07**
Word2Vec	46.03	47.98	53.92	48.83	51.04	57.21	58.21	61.05	63.40
FastText	54.00	61.09	64.61	63.40	64.16	67.36	66.43	69.60	70.13
FastText [147]	41.50	43.03	52.71	46.30	52.79	55.43	56.07	59.97	62.61

The individual embeddings evaluated the semantic, syntactic, relatedness, and word analogy tasks. Table 6.2 shows the 3COSADD (Add) and 3COSMULL (Mull) [145] performance. An analogy is solved correctly if the cosine similarity

Table 6.2: Semantic Syntactic and relatedness word analogy tasks performance summary

Models	Semantic(%)		Syntactic(%)		Relatedness(%)	
	Add	Mull	Add	Mull	Add	Mull
GloVe	**44**	**51**	**48**	**59**	**62**	**64**
FastText	39	43	41	49	58	62
Word2Vec	37	41	38	47	56	60
FastText[147]	33	37	34	39	51	57

of W_b (with W_b^*) and W_b^* (with the ground truth word) score difference is less than a predefined threshold value. The predefined threshold value is 3 for this dataset and selected by the similarity score simulations over the datasets.

(ii) WEC-based Individual Embedding Performance Various combinations of hyperparameters of three embedding techniques (e.g., GloVe, FastText, and Word2Vec) have generated 90 local contextual embedding models [18 for GloVe, 36 for FastText (Skip-gram & CBOW), 36 for Word2Vec (Skip-gram & CBOW)]. Intrinsic evaluators are used to evaluate a total of 90 models using syntactic and semantic similarity measures [58]. Based on the intrinsic evaluation performance, a total of 9 top-performing embedding models are selected for the authorship classification task. In particular, three models are chosen from GloVe, three from FastText, and three from Word2Vec embeddings based on the highest Pearson and Spearman correlation scores. Table 6.3 shows the best nine intrinsic evaluation results of three embedding techniques for 100 semantic and syntactic word pairs.

The maximum semantic word similarity in terms of Pearson (62.02%) and Spearman (63.66%) correlation scores has been achieved for the GloVe model with an embedding dimension (ED) of 250 and 13 windows size. In syntactic

Table 6.3: Summary of the intrinsic evaluation results for best-performing embedding techniques based on Spearman ($\hat{\rho}$) and Pearson ($\hat{r}$) correlations

			Semantic (%)		Syntactic (%)	
Model	ED	Window	$S_{s\hat{\rho}}$	$S_{s\hat{r}}$	$S_{y\hat{\rho}}$	$S_{y\hat{r}}$
	250	**13**	**62.02**	**63.66**	68.12	**72.30**
GloVe	**200**	**15**	56.33	57.77	**71.09**	72.01
	100	10	53.20	55.07	67.10	69.12
	250	13	57.11	58.19	44.34	46.31
FastText	200	15	53.64	53.38	39.47	38.31
	100	10	56.78	55.18	37.78	36.03
	250	13	44.31	45.10	31.78	30.51
Word2Vec	200	15	44.31	45.10	31.78	30.51
	100	10	44.31	45.10	31.78	30.51

similarity, the maximum Pearson correlation score (71.09%) is obtained from the Glove model with ED of 200 and 15 contextual window. In contrast, the highest Spearman correlation score (72.30%) is achieved for the GloVe with 250 ED and 13 window. The FastText model has achieved the second-best performance with Pearson and Spearman correlation for semantic and syntactic similarity evaluations. It is observed that the Word2Vec technique has achieved the lowest performance. Thus, the GloVe embedding technique with 250 embedding dimension and 13 contextual windows size performed better than Word2Vec and FastText.

(iii) CovEC-based Individual Embeddings Performance The intrinsic evaluation is carried out on a word-level semantic/syntactic performance. Table 6.4 shows the performance of Spearman ($\hat{\rho}$), Pearson ($\hat{r}$) and Cosine similarity of semantic (S_s), syntactic (S_y) and relatedness (R_s) datasets. The embedding parameter identification (EPI) Algorithm selects three embedding dimensions (EDs) (e.g., $ED \in \{200, 250, 300\}$) and two contextual windows (e.g., 12 and 13) for GloVe, FastText and Word2Vec methods. These three methods yield 18 single embedding models using $CovEC$ corpus. Table 6.4 illustrates the summary of the best-performed single embedding models.

Table 6.4: Intrinsic performance of the best-performed embedding models

Models	*Semantic* $S_s(\%)$			*Syntactic* $S_y(\%)$			*Relatedness* $S_r(\%)$		
	$S_{s\hat{\rho}}$	$S_{s\hat{r}}$	$\overrightarrow{C}_{S_m}$	$S_{y\hat{\rho}}$	$S_{y\hat{r}}$	$\overrightarrow{C}_{S_y}$	$S_{r\hat{\rho}}$	$S_{r\hat{r}}$	$\overrightarrow{C}_{S_r}$
GloVe	**65.97**	**67.10**	**79.13**	**70.93**	**76.33**	**80.41**	**81.67**	**81.89**	**88.10**
Word2Vec	49.74	52.07	56.92	51.50	54.29	60.80	60.11	63.19	66.28
FastText	56.29	63.48	67.03	66.11	67.16	67.20	68.84	72.59	74.31

In the intrinsic evaluation of embedding models presented in Table 6.4, the

minimum and maximum results for semantic similarity (S_s) include $s\hat{\rho}$ values of 49.74 (Word2Vec) and 65.97 (GloVe), respectively. For syntactic similarity (S_y), $y\hat{\rho}$ ranges from 51.50 (Word2Vec) to 70.93 (GloVe). In the case of relatedness similarity (S_r), $r\hat{\rho}$ varies from 60.11 (Word2Vec) to 81.67 (GloVe). Similarly, $\hat{r}$ and $\overrightarrow{C}$ values exhibit minimum and maximum performance results across the three evaluated embedding models.

Table 6.5 shows the performance of analogy tasks for single embedding models. In the analogy task performance evaluation detailed in Table 6.5, the minimum

Table 6.5: Performance of the best-performed embedding models for analogy tasks regarding 200 semantic, 200 syntactic & 200 relatedness word-pairs.

Models	Semantic $A_t(\%)$		Syntactic $A_t(\%)$		Relatedness $A_t(\%)$	
	Add	Mull	Add	Mull	Add	Mull
GloVe	**46**	**52**	**50**	**60**	**64**	**66**
FastText	42	44	42	50	60	64
Word2Vec	38	42	40	48	60	62

and maximum results across different categories are as follows: For semantic analogy tasks, the minimum performance (A_t) includes 38 (Word2Vec) and 46 (GloVe), while the maximum performance is 52 (GloVe) and 42 (FastText) for "Add" and "Mull" analogy types, respectively. In syntactic analogy tasks, "Add" analogies exhibit a range of 40 (Word2Vec) to 50 (GloVe), while "Mull" analogies range from 48 (Word2Vec) to 60 (GloVe). For relatedness analogy tasks, the minimum and maximum performances are 60 (Word2Vec) and 64 (GloVe) for "Add" analogies and 62 (Word2Vec) and 66 (GloVe) for "Mull" analogies, respectively. These results demonstrate variations in analogy task performance among different embedding models.

6.4.2 Performance of Meta-Embeddings

The Meta-Embedding models are generated for the BEC corpus. The BEC corpus generates different types of meta-embedding models. The meta-embedding performances are described in the following paragraphs.

(i) BEC-based Meta-Embedding Performance: The Meta-Embedding generates nine ME-based models (3 for AVG-M, 3 for CONC-M & 3 for CONC-SVD-M), and the results of the three best-performed models are shown in Table 6.6. Among all the embedding models, AVG-M achieved the maximum Spearman, Pearson correlation (for semantic, syntactic and relatedness datasets), whereas

CONC-SVD-M models achieved maximum Spearman and average cosine similarity of 82.20% and 86.21%. The ME-based analogy task performance is shown in

Table 6.6: Spearman ($\hat{\rho}$), Pearson ($\hat{r}$) and average cosine similarity (A_{CS}) statistics for 200 semantic, syntactic & relatedness word pairs. GloVe & FastText (Skip-gram) features are used in AVG-M, CONC-M & CONC-SVD-M. CONC-M embedding dimension of 600 & others is of 300.

Models	Semantic(%)			Syntactic(%)			Relatedness(%)		
	$S_{s\hat{\rho}}$	$S_{s\hat{r}}$	A_{CS}	$S_{y\hat{\rho}}$	$S_{y\hat{r}}$	A_{CS}	$S_{r\hat{\rho}}$	$S_{r\hat{r}}$	A_{CS}
AVG-M	**65.12**	**66.98**	**78.20**	**71.31**	**76.01**	**79.10**	81.21	**82.01**	86.06
CONC-M	64.98	65.60	77.03	70.69	75.40	79.04	80.21	81.20	85.72
CONC-SVD-M	65.01	65.67	76.89	70.04	75.05	78.87	**82.20**	81.72	**86.21**

Table 6.7. The AVG-M embedding model solved a maximum of 54% semantic analogy tasks, and the AVG-M Meta-Embedding solved 60% syntactic analogy tasks. For the other cases, the CONC-SVD-M Meta-Embedding model correctly answered 66% relatedness analogy tasks using 3COSMULL (Mull). The analogy task performance is not up to the mark due to out-of-vocabularies issues.

Table 6.7: Semantic Syntactic and relatedness word analogy tasks performance summary. GloVe & FastText (Skip-gram) features are used in AVG-M, CONC-M & CONC-SVD-M. CONC-M embedding dimension of 600 & others is of 300.

Models	Semantic(%)		Syntactic(%)		Relatedness(%)	
	Add	Mull	Add	Mull	Add	Mull
AVG-M	**48**	**54**	**51**	**60**	63	**66**
CONC-M	45	52	49	58	60	65
CONC-SVD-M	46	48	50	59	**64**	**66**

The intrinsic evaluation summarizes that the proposed Meta-Embedding (e.g., AVG-M) gained the maximum performance in Spearman, Pearson, cosine similarity, and analogy performance. The Meta-embedding models achieve better accuracy concerning individual embedding because of features fused from different embedding techniques, reducing the feature outliers, variance, and standard deviations.

6.4.3 Performance of Attention-based Meta-Embeddings:

In this study, we have generated and evaluated the attention-based meta-embedding for the CovEC corpus and intrinsically selected the best-performing Covid text feature representation model for Covid text identification purposes.

The intrinsic evaluation is carried out on a word-level semantic/syntactic performance. Therefore, the position encoding value can not be used in attention

calculation. Only the attention and fusion operations are employed to represent word semantics. Table 6.8 shows the performance of Spearman ($\hat{\rho}$), Pearson ($\hat{r}$) and Cosine similarity of semantic (S_s), syntactic (S_y) and relatedness (S_r) datasets. The embedding parameter identification (EPI) Algorithm selects three embedding dimensions (EDs) (e.g., $ED \in \{200, 250, 300\}$) and two contextual windows (e.g., 12 and 13) for GloVe, FastText and Word2Vec methods. These three methods yield 18 single embedding models using $CovEC$ corpus. The best-performed embedding models are used to generate the attention-based feature fusion model using Concatenation ($ConCat$), Averaging (Average), and Concatenation with principal component analysis ($ConCat - PCA$) methods [148]. The $ConCat$ method produced four fused embedding feature matrices (e.g., GloVe+FastText, GloVe+Word2Vec, FastText+Word2Vec, GloVe+FastText+Word2Vec). The other two methods also generated eight fused embedding feature matrices. Among these 18 single and 12 fused embedding models, three single (e.g., one from GloVe, one from FastText and one from Word2Vec) embedding and three fused embedding (e.g., APeCGF: Attention-based embedding with ConCat (GloVe, FastText), APeAGF: Attention-based embedding with Averaging (GloVe, FastText), APeCpGF: Attention-based embedding with ConCat-PCA (GloVe, FastText)) models are selected for the downstream task (i.e., CTI). Table 6.8 illustrates the summary of the best-performed single and fusion-based embedding models.

Table 6.8: Intrinsic performance of the best-performed attention-based meta-embedding models

Models	*Semantic* $S_s(\%)$			*Syntactic* $S_y(\%)$			*Relatedness* $S_r(\%)$		
	$S_{s\hat{\rho}}$	$S_{s\hat{r}}$	$\overrightarrow{C}_{S_m}$	$S_{y\hat{\rho}}$	$S_{y\hat{r}}$	$\overrightarrow{C}_{S_y}$	$S_{r\hat{\rho}}$	$S_{r\hat{r}}$	$\overrightarrow{C}_{S_r}$
APeAGF	**68.20**	**69.10**	**81.78**	**73.68**	**79.27**	**82.41**	83.01	**84.70**	88.59
APeCGF	65.83	67.04	78.90	72.93	77.46	81.18	82.21	83.57	87.02
APeCpGF	66.70	67.96	79.02	73.05	77.53	80.11	**83.79**	83.72	**88.52**

The maximum Spearman ($\hat{\rho}$), Pearson ($\hat{r}$), and average cosine similarity ($\overrightarrow{C}_{S_m}$) of semantic dataset are 68.20%, 69.10% and 81.78% respectively achieved by APeAGF. Similarly, the syntactic dataset obtained the maximum accuracy of 73.68%, 79.27% and 82.27% by APeAGF. In contrast, the relatedness dataset obtained the maximum value for Spearman ($\hat{\rho}$) and Pearson ($\hat{r}$) from APeCpGF. Overall, Pearson ($\hat{r}$) performance has an improvement of 2.94% for the syntactic dataset using the attention-based feature fusion embedding model compared to the single embedding (i.e., GloVe, FastText & Word2Vec). The attention operation improves the word-word correlations, whereas the feature fusion operation

combines the unique features of semantic, syntactic, and relatedness from the single embedding. Thus, it is confirmed that attention-based feature fusion is better than single embedding for extracting textual features.

Table 6.9 shows the performance of analogy tasks for attention-based meta-embedding models. In most cases, the intrinsic evaluation revealed that the

Table 6.9: Performance of the best-performed attention-based meta-embedding models for analogy tasks regarding 200 semantic, 200 syntactic & 200 relatedness word-pairs.

Models	Semantic $A_t(\%)$		Syntactic $A_t(\%)$		Relatedness $A_t(\%)$	
	Add	Mull	Add	Mull	Add	Mull
APeAGF	**50**	**56**	**54**	**64**	62	**68**
APeCGF	48	54	52	60	62	66
APeCpGF	**50**	50	52	62	**66**	**68**

attention-based average feature fusion (APeAGF) with GloVe+FastText achieved the highest performance for semantic, syntactic, and relatedness datasets. The attention-based fused feature combines the morphologically significant variations of the Bengali language. The maximum semantic (50% & 56%) and syntactic (54% & 64%) analogy accuracies have been achieved from APeAGF feature fusion, whereas the relatedness dataset obtained a maximum accuracy of 68% for APeCpGF and APeAGF feature fusions.

Among 30 embedding models (18 for single and 12 for fusion), the intrinsic evaluators select the best three models to perform the downstream task (CTI). Thus, instead of sending all 30 models for training, the system can use only the best models, reducing the downstream task's time and storage complexity. Due to intrinsic evaluation, 90% (i.e., only the top three models can be used instead of 30 for CTI task evaluation) of training time was saved to perform CTI tasks. For better clarification, we investigate one best single embedding model (GloVe) and two fused embedding models (e.g., APeAGF and APeCpGF) for CTI tasks.

Findings of Intrinsic Performance: In this research, we initially employed three individual embedding techniques, i.e., GloVe, FastText, and Word2Vec, using our self-developed corpora. Through intrinsic evaluation (as shown in 6.1, 6.2, 6.3, 6.4, and 6.5), we determined that GloVe embedding outperformed the other embeddings in terms of intrinsic performance. To address context-related limitations and out-of-vocabulary (OOV) issues, we subsequently implemented Meta-Embedding and Attention-based Meta-Embedding. These techniques involved averaging, concatenation, and concatenation with singular value decomposition (SVD) to fuse features from individual embeddings. After rigorous intrinsic eval-

uation, specifically referring to Tables 6.6, 6.7, 6.8, and 6.9, it was found that the average Meta-Embedding (AVG-M) and the Attention-based positional embedding with averaging fusion of GloVe and FastText (APeAGF) achieved the best performance for downstream text classification tasks. These embeddings, namely GloVe, AVG-M, and APeAGF, were selected for text classification purposes.

6.5 Performance of Text Classifiers: Extrinsic Evaluation

There are four deep learning-based models are achieved the best performance in this research, such as (i) GloVe with CNN, (ii) GloVe with VDCNN, (iii) Meta-Embedding with CNN, and (iv) Attention-based Meta-Embedding with ACNN. In the following each of the modules.

6.5.1 Performance of GloVe with CNN Model

Intrinsic evaluation selects the top *nine* best-performing embedding models. The authorship classification task is evaluated on BACC-18, BAAD16, and LD. To investigate the Bengali authorship classification task performance, we implemented 36 models using the combinations of 9 embedding models and four classification techniques (SVM, SGD, CNN & LSTM). In particular, 36 models are generated from four classification techniques where SVM and SGD contributed to 18 models ([9 (embedding model) x 2 (SVM & SGD)]), 9 models contributed to CNN-based models ([9 (embedding model) x 1 (CNN)]) and LSTM-based models contributed to 9 models. Among 36 models, Table 6.10 illustrates 18 top-performing models, which are selected based on different classification techniques. The CNN + GloVe model achieved the highest accuracy of 86.98% (for BACC-18), 91.08% (for BAAD16), and 96.67% (for LD) in 100 EM. In the 200 ED, BACC-18 test datasets obtained the highest of 91.07% accuracy using GloVe+CNN 200 ED and 92.96% accuracy for BAAD16 test datasets, whereas Fasttext+CNN gained the maximum of 98.53% accuracy for LD test datasets. With the 250 embedding dimension, the GloVe+CNN obtained the highest accuracy of 93.45% (for BACC-18), 95.02% (for BAAD16) and 98.67% (for LD). The analysis revealed that the proposed approach (GloVe+CNN) achieved the best performance in all datasets (BACC-18: 93.45%, BAAD16: 95.02% & LD: 98.67%) with 250 ED. Due to the superiority of the Glove+CNN model, the subsequent analysis is performed with the 250 feature dimension only.

Table 6.10: Macro-F1 (M_{ap}), weighted-F1 (WA_p) and Accuracy (A) measures on three datasets

		BACC-18 (%)			BAAD16 (%)			LD (%)		
Model	ED	M_{ap}	WA_p	A	M_{ap}	WA_p	A	M_{ap}	WA_p	A
GloVe+CNN		**81.00**	**87.00**	**86.98**	**87.00**	**91.00**	**91.08**	**97.00**	**96.00**	**96.67**
GloVe+SVM		61.00	64.00	64.21	64.00	68.00	68.37	94.00	95.00	94.66
Fasttext+CNN		80.00	85.00	85.04	85.00	89.00	88.79	97.00	96.00	96.67
Fasttext+SVM	100	62.00	64.00	64.10	65.00	70.00	69.88	95.00	95.00	94.67
Fasttext+SGD		63.00	67.00	66.95	65.00	71.00	71.17	95.00	95.00	94.67
Word2Vec+CNN		78.00	82.00	81.89	83.00	86.00	86.03	96.00	96.00	96.67
Word2Vec+SVM		58.00	61.00	60.97	62.00	67.00	67.06	93.00	93.00	93.33
Word2Vec+SGD		59.00	60.00	60.02	64.00	69.00	69.19	95.00	95.00	94.67
GloVe+SGD		62.00	66.00	65.89	65.00	70.00	70.15	95.00	95.00	94.66
[149]										
GloVe+CNN		**84.00**	**91.00**	**91.07**	**89.00**	**93.00**	**92.96**	**98.00**	**98.00**	97.60
Fasttext+CNN	200	82.00	90.00	89.97	88.00	92.00	92.03	98.00	**99.00**	**98.53**
Word2Vec+CNN		80.00	88.00	88.06	86.00	91.00	90.74	98.00	98.00	97.33
GloVe+CNN		**87.00**	**93.00**	**93.45**	**93.00**	**95.00**	**95.02**	**99.00**	**99.00**	**98.67**
GloVe+LSTM		85.00	90.00	89.90	89.00	92.00	92.02	97.00	98.00	98.13
Fattext+CNN	250	86.00	91.00	90.94	92.00	94.00	94.11	**99.00**	**99.00**	98.40
Fasttext+LSTM		85.00	91.00	91.09	90.00	93.00	93.12	98.00	98.00	98.13
Word2Vec+CNN		83.00	89.00	89.20	89.00	91.00	91.16	98.00	98.00	97.33
Word2Vec+LSTM		82.00	89.00	89.09	90.00	91.00	90.93	98.00	98.00	97.60

Table 6.11 shows the statistical summary of the GloVe+CNN model concerning 250 embedding dimensions. Author-wise classification performance is denoted by Precision, Recall, and Micro F1-score. Results indicate that for BACC-18, author 10 obtained the highest precision (99.00%) and F1-score (98.00%), whereas author 13 achieved the lowest recall (62.00%) and F1-score (75.00%). In the BAAD16 dataset, the highest recall (99.00%) and F1-score (97%) are achieved for author 07. For author 21, the approach obtained the highest precision, whereas author 03 achieved the lowest recall (64.00%) and F1-score (75.00%). On LD, a maximum of 100.00% performance measures (precision, recall, and Micro F1-score) are obtained for one author (01), whereas a lowest of 98.00% is achieved for authors 02 and 10, respectively. However, the average accuracy of the proposed method on the LD dataset is 98.67%. The LD dataset contains only three authors with a small training/testing set where huge stylometric feature variations are observed between author 01 and the other two authors (02 & 10). Usually, most of the literature has been written by author 01 in Shadhu-bhasha, whereas authors 02 and 10 have written in Cholito-bhasha. These stylists' variations carry a huge impact on feature representation, and thus, author 01 achieved better accuracy than others. The proposed GloVe+CNN architecture can detect these distinguishable features of the author's (01) text very well, which helps to predict all the texts in the test set correctly.

Figure 6.1 summarizes the performance of the proposed model (GloVe+CNN with 250 EM) on three datasets. The bar denotes the average precision, recall, and Micro F1-score for BACC-18, BAAD16, and LD. The proposed model per-

Table 6.11: Statistical measures summary of GloVe+CNN with 250 ED on three datasets).

Datasets	Author code	P(%)	R(%)	M_{ap} (%)	support
BACC-18	01	98.00	94.00	96.00	179
	17	97.00	82.00	89.00	76
	19	78.00	90.00	84.00	20
	03	90.00	76.00	83.00	84
	02	94.00	97.00	96.00	720
	08	92.00	65.00	76.00	34
	09	90.00	95.00	92.00	57
	10	**99.00**	97.00	**98.00**	461
	11	91.00	91.00	91.00	47
	12	90.00	97.00	93.00	196
	18	**72.00**	**100.00**	84.00	42
	24	98.00	70.00	82.00	61
	05	84.00	70.00	77.00	61
	15	95.00	98.00	96.00	608
	16	81.00	**100.00**	90.00	22
	14	84.00	72.00	78.00	58
	07	84.00	98.00	90.00	97
	13	95.00	**62.00**	**75.00**	32
BAAD16	06	97.00	94.00	95.00	220
	01	93.00	88.00	90.00	112
	07	96.00	**99.00**	**97.00**	906
	04	95.00	78.00	86.00	93
	03	96.00	**61.00**	**75.00**	44
	22	98.00	95.00	96.00	95
	02	**91.00**	92.00	92.00	252
	23	95.00	94.00	95.00	210
	08	97.00	98.00	**97.00**	282
	25	94.00	92.00	93.00	177
	10	93.00	98.00	95.00	261
	11	94.00	98.00	96.00	169
	05	94.00	98.00	96.00	393
	20	94.00	92.00	93.00	155
	14	95.00	95.00	95.00	186
	21	**100.00**	78.00	88.00	37
LD	**01**	**100.00**	**100.00**	**100.00**	250
	02	**98.00**	**98.00**	**98.00**	250
	10	**98.00**	**98.00**	**98.00**	250

formed slightly better in LD than BACC-18 and BAAD16 (e.g., precision values increased 9.00% and 5.00% compared to BACC-18 and BAAD16). Due to the smaller number of authors (only three), LD has achieved a slightly better performance than BACC-18 (18 authors) and BAAD16 (16 authors). The maximum

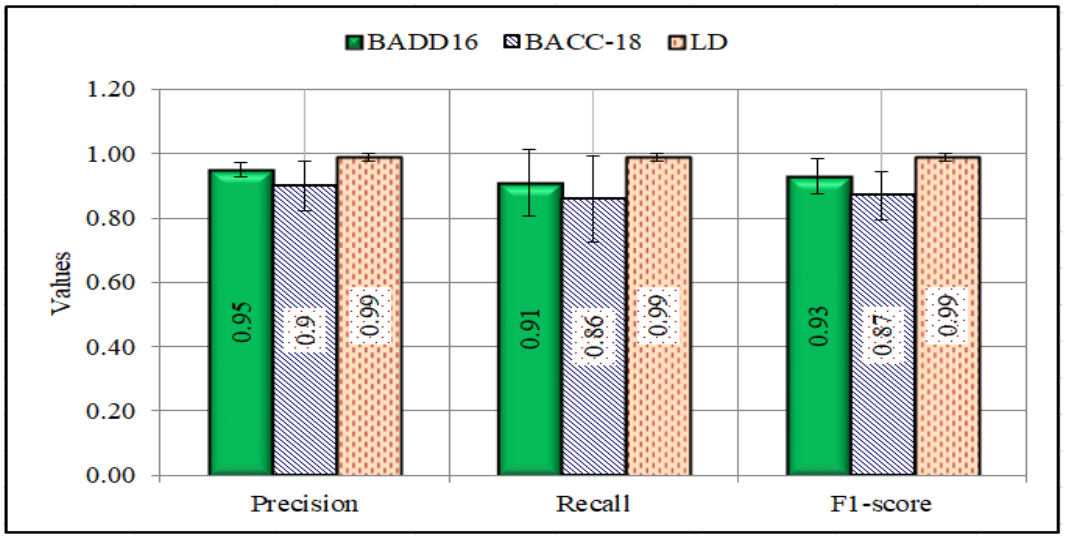


Figure 6.1: Performance of GloVe+CNN model on BACC-18, BAAD16 and LD. The error bar indicates the standard error.

Table 6.12: Performance comparison of authorship attribution

	Accuracy (%)		
Model	BAAD16	BACC-18	LD
FastText (Char)+CNN [99]	79.44	76.82	97.60
Bigrams-TF+NB [128]	65.17	62.44	98.00
GloVe+SVM [150]	68.00	64.21	94.66
Proposed (GloVe+CNN)	**95.02**	**93.45**	**98.67**

7.6% precision, 13.43% recall, and 7.54% Micro F1-score error are obtained on BACC-18 datasets. The minimum 1.15% precision, recall, and Micro F1-score are achieved on LD.

Comparison with previous approaches Table 6.12 shows the performance comparison concerning the proposed system's accuracy with the previously developed methods ([99], [128]), [150]) on BACC-18, BAAD16, and LD datasets.

Results indicate that the proposed method achieved the highest accuracy of 98.67% than presented in [99] (97.6%), [150] (94.66%) and [128] (98%) for LD. Concerning BAAD16, the proposed technique achieved a maximum accuracy of 95.2%, which is better than Khatun et al. [99] (79.44%), Lahiri et al. [150] (68%) and Phani et al. [128] (65.17%). Moreover, in the case of BACC-18, the proposed method obtained the highest accuracy of 93.46%, whereas the method in [99] obtained 76.82% and [150] gained 64.21%. Thus, for three datasets, the proposed method achieved the highest accuracy and outperformed the existing techniques.

Comparison with Transformer-based Techniques In recent years, transformer-based approaches have gained increased attention among NLP researchers for various text classification tasks. This work carried out a set of new experiments using transformer-based techniques on the three corpora: BACC-18, BAAD16, and LD. Although many variants of transformer-based techniques are available, this work investigated two popular methods (m-BERT [16] and Distil-BERT [26]) for the authorship classification task. The pre-trained m-BERT and Distil-BERT hugging face models[1] [2] are used to evaluate the performance of the authorship classification. All of the hyperparameters are used as default. Table 6.13 shows the performance of m-BERT and Distil-BERT models including the proposed model on the three datasets.

Table 6.13: Performance of transformer models in authorship classification

Datasets	Techniques	Accuracy (%)
BACC-18	M-BERT	92.46
	Distil-BERT	91.06
	Proposed (GloVe+CNN)	**93.45**
BAAD16	M-BERT	94.09
	Distil-BERT	93.54
	Proposed (GloVe+CNN)	**95.02**
LD	M-BERT	98.00
	Distil-BERT	97.33
	Proposed (GloVe+CNN)	**98.67**

Results indicate that m-BERT achieved a higher accuracy of 1.4% than Distil-BERT for the BACC-18 dataset due to the better semantic feature representation at word label with a lower 40% trainable parameters[3].

On the other hand, the m-BERT model achieved a lower accuracy than the proposed method for most cases. Specifically, the m-BERT model showed an approximately 0.99% lower accuracy compared to the proposed technique. Bengali author's texts usually are available in two morphological variants, such as Sadhu-bhasha and Cholito-bhasha. These variations are not considered in the pre-train transformer-based models (i.e., m-BERT, Distil-BERT). Thus, the pre-trained models suffer from handling the author's text written in Shadhu-bhasha form (such as authors 01 and 10, usually written in Shadhu-bhasha). However, the developed corpus contained authors' texts of both forms. The previous study also revealed that the pre-trained transformer-based models do not perform well for the low-resource language [151], including Bengali, due to its limited unique

[1]https://huggingface.co/bert-base-multilingual-cased,
[2]https://huggingface.co/transformers/model_doc/distilbert.html
[3]https://huggingface.co/transformers/model_doc/distilbert.html jsdgkjdkaka

words, out-of-vocabulary problem, and incapability to handle Bengali morphological variants. Although further investigations with other transformer-based techniques (such as Bangla-BERT, XLM, and ELECTRA) are deemed necessary, the preliminary results revealed that the proposed model outperformed other techniques, including transfer-based techniques (m-BERT and Distil-BERT).

6.5.2 Performance of GloVe with VDCNN Model

To investigate the Bengali text classification task performance, 109 models are implemented using the combinations of embedding models and classification techniques (ML, CNN, sequential, BERT). In particular, 108 models are generated from three classification techniques where three ML-based techniques contributed to 36 models [12(embedding model) × 3 (SVM, Libsvm & SGD)], three CNN-based techniques contributed to 36 models [12(embedding model) × 3 (CNN, DCNN, VDCNN)] and three sequential-based technique generated 36 models [12(embedding model) × 3 (LSTM, GRU, GRU+LSTM)]. One model is generated from the BERT-based technique (m-BERT). Among 109 models, Table 6.14 illustrates the result of 26 top-performing models, which are selected based on different classification techniques (8 for ML, 8 for CNN, 9 for sequential, 1 for transformer-based). Among 8 ML models, GloVe+SGD gained the maximum

Table 6.14: Bengali text classification task performance of26 best-performed models

Techniques	Models	ED	CW	F_1-score	A(%)
Traditional ML	TF-IDF+SVM	-	-	77.24	78.17
	Word2Vec (Skip-gram)+Libsvm	300	9	81.40	82.06
	GloVe+Libsvm	300	9	82.10	83.15
	FastText (Skip-gram)+Libsvm	300	9	81.89	82.72
	TF-IDF+SGD	300	9	79.31	80.41
	Word2Vec (Skip-gram)+SGD	300	9	84.15	85.31
	GloVe+SGD	300	9	**85.08**	**86.20**
	FastText (Skip-gram)+SGD	300	9	84.78	85.98
Convolutional NN	GloVe+CNN	200	12	89.41	90.82
	Word2Vec (Skip-gram)+CNN	300	9	89.97	91.23
	FastText (Skip-gram)+CNN	300	9	90.06	91.79
	Word2Vec (Skip-gram)+DCNN	300	9	92.18	93.23
	GloVe+DCNN	300	9	93.50	94.86
	FastText (Skip-gram)+DCNN	300	9	92.69	93.85
	Proposed(GloVe+VDCNN)	300	9	**97.00**	**96.96**
	GloVe+DCNN	200	12	92.50	93.16
Sequential	Word2Vec (Skip-gram)+LSTM	300	9	91.98	92.15
	GloVe+LSTM	200	12	92.30	92.80
	GloVe+CNN-GRU	200	12	89.47	90.07
	GloVe+LSTM	300	9	94.14	94.76
	GloVe+CNN-LSTM	300	9	**95.14**	**95.20**
	FastText (Skip-gram)+LSTM	300	9	93.40	93.97
	FastText (Skip-gram)+CNN-LSTM	300	9	94.15	94.88
	Word2Vec (Skip-gram)+CNN-LSTM	300	9	92.73	93.00
	GloVe+CNN-LSTM	200	12	92.95	93.12
Transformer	**m-BERT**	768	-	-	**92.45**

accuracy of 86.20% (F_1-score=85.08%) to perform the Bengali text classification task. The SVM and Libsvm classification techniques cannot generate the separable decision boundary in the developed training corpus ($BDTC$) due to the lack of generalized text features and word inflection rate. Thus, the performance of SVM and Libsvm-based models showed lower accuracy. Out of 36 CNN-based models, the results of *eight* best-performed models are shown in Table 6.14. The results showed that the GloVe+VDCNN achieved the highest accuracy of 96.96% (with an average F_1-score of 97.00%) than other CNN-based combinations. The CNN is implemented using a single-layer multi-kernel [152], and this technique suffers from capturing adequate semantic and syntactic features (at the sentence and document levels). As a result, the CNN-based models showed a quite lower accuracy compared to DCNN and VDCNN-based models. Among 10 best performing sequential-based models, the GloVe+CNN-LSTM gained the maximum accuracy of 95.69% (F_1-score= 95.14%). The GRU contains fewer trainable parameters concerning LSTM. These fewer parameters cannot extract all relevant features from the large text datasets (e.g., $BDTC$). As a result, the GRU-based model did not perform well using the developed corpus (the highest accuracy of 90.07% is obtained for CNN-GRU). This work investigates the Bengali text classification task performance using a transformer-based technique (m-BERT) on the developed corpus. The pre-trained m-BERT model achieved 92.45% accuracy. The pre-trained m-BERT model is well-performed for the English language, but the performance of this pre-rained model is not satisfactory for the low resource language (e.g., Italian [151]), including Bengali.

The results revealed that the proposed model (GloVe+VDCNN) obtained the maximum accuracy of 96.96% among all models. The traditional feature extractors (TF, IDF & TF-IDF) are not able to capture the semantic and syntactic features, whereas the GloVe, Word2Vec (Skip-gram), FastText (Skip-gram), and m-BERT can capture these types of features. Also, ML classifiers cannot capture the local (e.g., word & sentence level) and global (e.g., document level) semantics. In contrast, the CNN, DCNN, VDCNN, LSTM, GRU, and m-BERT can carry the local and global semantics. The Bengali language has morphological variations in textual form: Sadhu-bhasha and Cholito-bhasha. The morphological variations like Sadhu-bhasha and Cholito-bhasha are not considered in the pre-train model. The m-BERT model pre-trained the Wikipedia Bengali dataset, which suffers from handling the variations of Cholito-bhasha and Cholito-bhasha of Bengali. However, the proposed GloVe model covers these morphological variations with the most extensive vocabulary sets of the developed corpus. Thus, the GloVe+VDCNN model obtained better performance than pre-trained m-BERT.

On the other hand, Word2Vec (Skip-gram) and FastText (Skip-gram) only considered the contextual windows, and these models cannot cover the whole corpus and word pair statistics. However, the GloVe model has taken into consideration the entire corpus and word pair statistics. The GloVe model reduces the feature vector values for high-frequency words, but FastText (Skip-gram) and Word2Vec do not consider these characteristics. In summary, the experimental analysis revealed that the proposed method (GloVe+VDCNN) achieved the maximum accuracy (96.96%) than all other models due to its better feature extraction capability and hyperparameters selection strategy. A recent study on embedding has also shown that the FastText (Skip-gram), Word2Vec (Skip-gram), and pre-trained m-BERT cannot be performed well for the Italian language (a low-resource language) [151].

Statistical Evaluation Table 6.15 presents the result of the several statistical measures (i.e., precision, recall, F_1-score measure) of the Bengali document classifier. The testing data set contains $21,477$ Bengali text documents exclusive to the train and validation data sets. Here, the term support represents the number of documents.

Table 6.15: Statistical evaluation of document classifier

Document Category	P	R	F_1-score	Support
sports	0.99	0.98	0.99	1,958
lifestyle	0.98	0.99	0.99	2,017
art	0.78	0.98	0.87	236
science & technology	0.97	0.92	0.95	1,183
education	0.96	0.98	0.97	970
environment	0.95	0.92	0.93	534
entertainment	0.98	0.98	0.98	3,439
economics	0.94	0.96	0.95	953
health	0.99	0.97	0.98	859
opinion	0.96	0.93	0.94	1,985
politics	0.97	0.97	0.97	3,713
crime	0.97	0.98	0.98	2,814
accident	0.98	0.98	0.98	816
avg/total	0.97	0.97	0.97	21,477

Two categories *lifestyle* and *sports* achieved the highest f_1 score, which is close to 0.99. Only one category (*art*) achieved the lowest F_1-score and precision values (F_1-score= 0.87, and $p = 0.78$) due to the failure of classification of few large documents. Other categories provided fairly good classification scores.

Performance Comparison: In recent years, few research activities have been carried out on text classification in Bengali. Due to the unavailability of other authors' datasets and benchmark corpus, the contemporary techniques are implemented on the developed corpus (BDTC), including existing methods ([4, 96, 93, 94, 97, 98]). The performance of the proposed system is compared with the previous approaches in terms of corpus size and accuracy. Table 6.16 presents the number of training documents, testing documents, and classes used in the existing approaches and the proposed method.

Table 6.16: Summary of previous Bengali document classification techniques concerning training size, testing size, and classes

Approaches	#Training Data	#Testing Data	#Class
GloVe+DCRNNs [4]	120,000	36,207	12
Word2Vec+K-NN+SVM [96]	19,750	4,713	7
TF-IDF+SGD [93]	5,448	3,679	9
Word2Vec+SGD [94]	10,000	4,651	9
Word2Vec+DCNN [97]	86,199	10,707	12
Graph-based+LSTM-RNN [98]	-	14,373	9
Proposed (GloVe+VDCNN)	**106,899**	**21,477**	**13**

Table6.16 indicates that the proposed approach is developed on the largest training and testing datasets (106, 899 and 21, 477) with a higher number of classes (13 classes) than the existing approaches of Bengali document categorization.

Table 6.17 summarizes the comparison of text document categorization systems in Bengali.

Table 6.17: Summary of the performance comparison

Approaches	Accuracy (%)
GloVe+DCRNNs[4]	95.89
Word2Vec + Libsvm [153]	82.06
GloVe + Libsvm	83.51
TF-IDF+SGD [93]	80.41
Word2Vec + SGD [94]	85.31
GloVe + SGD	86.20
Word2Vec + DCNN [97]	93.23
GloVe + DCNN	94.86
Proposed (GloVe + VDCNN)	**96.96**

Results indicated that the accuracy of Libsvm-based classifiers gained accuracy between 78.17% and 83.51% on the developed *BDTC* dataset. The accuracy of the SGD-based classifications varied from 80.41% to 86.20%. DCNN-based classifier achieved the maximum accuracy of up to 94.86%. This research

compares all the previous approaches and other methods (such as Word2Vec + Libsvm, Glove + Libsvm, Glove + SGD, and Glove + DCNN), which leads to the observation that the proposed VDCNN-based technique achieved significantly higher accuracy of 96.96% outperforming all the approaches.

Usually, the performance of the classification model heavily depends on the number of training samples and classes [154]. When the number of classes increases, the classification technique faces inseparability problems [97]. Thus, the classification model starts losing its uniqueness and degrades its accuracy. Moreover, the classification model relies on better data distributions while training with more samples. These helped the classification model to improve accuracy with various distributions of the large corpus. Although further investigations with different corpus sizes and classes are deemed necessary, the preliminary results in Table 6.17 revealed that the proposed model's performance (i.e., accuracy) increased with the lower number of document classes and with a higher number of training samples.

Human Experts vs. Proposed Technique: The performance of the proposed system of Bengali document categorization is compared with the human experts (HEs) in terms of accuracy. Three experts who are postgraduates in Bengali linguistics are asked to manually classify the 21,477 text documents into one of 13 categories. A single document is read by three experts individually and classified accordingly. A document that achieved the highest number of category counts is selected as the final category. The accuracy of the human experts is calculated by the ratio of the total number of documents that predicted the class correctly and the total number of test documents. The accuracy of the proposed system is determined from the confusion matrix by the ratio of correctly classified count and total documents. For example, in the case of the health category, the system correctly predicts 833 documents out of 859 documents which produced the accuracy of 96.97% (833/859) (See in Fig. 6.15). Table 6.18 shows a summary of the comparison.

ANOVA (Analysis of Variance) analysis shows that there are no significant differences in accuracy achieved by the human experts while assigning the categories of text documents ($F(2, 24) = 0.959, p = 0.39, \eta^2 = 0.050$). Moreover, the average accuracy of the human experts 1, 2, and 3 are 99.5%, 99.7%, and 97.3%, respectively. The analysis of comparison shows that human experts are better at identifying documents than the proposed method (99.49% vs. 96.96%). Although the proposed system shows almost similar performance as human experts in identifying lifestyle and art categories, there are significant differences

Table 6.18: Performance comparison between human experts and proposed System

Category Name	Human-Experts Accuracy(%) (correctly predicted)	VDCNN Accuracy(%) (correctly predicted)	Support
sports	100.00 (1,958)	98.26 (1,924)	1,958
lifestyle	100.00 (2,017)	98.86 (1,994)	2,017
art	97.46 (230)	98.31 (232)	236
science & technology	100.00 (1,183)	92.10 (1,089)	1,183
education	100.00 (970)	98.15 (952)	970
environment	100.00 (534)	91.76 (490)	534
entertainment	100.00 (3,439)	98.10 (3,373)	3,439
economics	100.00 (953)	96.43 (919)	953
health	98.95 (850)	96.97 (833)	859
opinion	98.18 (1,949)	93.00 (1,846)	1,985
politics	98.79 (3,668)	97.15 (3,607)	3,713
crime	100.00 (2,814)	98.44 (2,770)	2,814
accident	100.00 (816)	97.55 (796)	816
avg / total	99.55 (21,381)	96.96 (20,825)	21,477

for other categories (such as crime, environment, opinion, and science & technology, respectively). ANOVA analysis also demonstrated that there is a substantial statistical difference in accuracy measures between human experts and VDCNN methods ($F(1, 12) = 16.07, p = 0.0017, \eta^2 = 0.39$). Fig. 6.2 also illustrates this result.

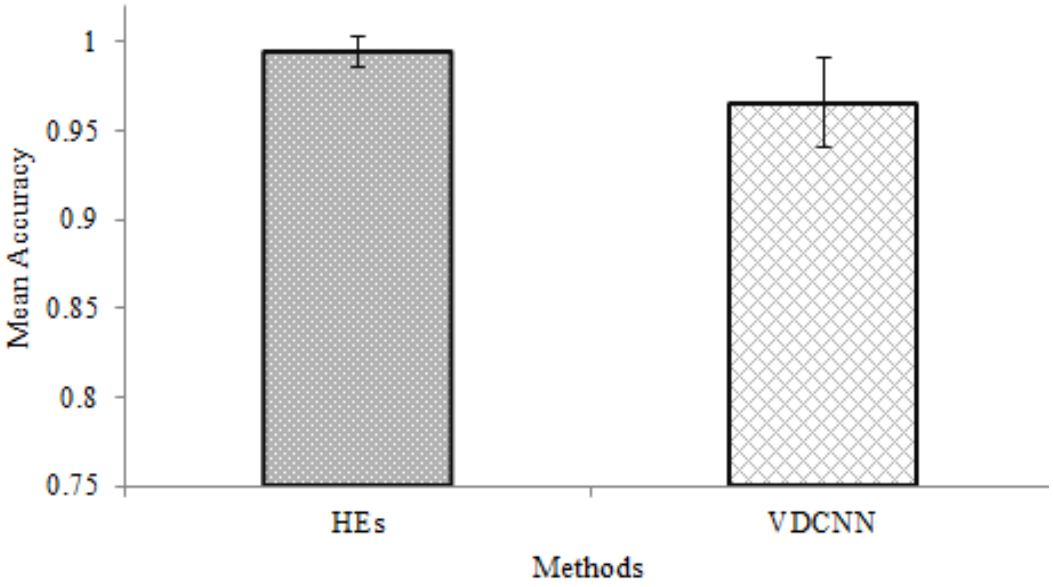


Figure 6.2: Mean accuracy obtained by the human experts and VDCNN. Error bar indicates the standard deviation

The performance of the proposed system can be improved by adding more distinct data distribution in the lower support categories and extracting more common semantic and syntactic features.

6.5.3 Performance of Meta-Embedding with CNN Model

The proposed single-layer multi-kernel CNN model is tuned using the AHST algorithm (e.g., algorithm 9) and Keras Tuner [134]. The initial, tuned, and Keras Tuner accuracy is shown in Table 6.19. The maximum accuracy of 90.07±.001% is achieved from AHST-tuned hyperparameters and the minimum from initial hyperparameters. The parameters are tuned on GTX 1070 GPU with Core-i7 processor and 32 GB physical memory. These tuned parameters are used for another three corpora.

Table 6.19: Summary of CNN performance with initial, tuned & KerasTuner using corpus BTCC11

Method	Accuracy(%)
CNN with initial hyperparameters	85.14±.001%
CNN with tuned hyperparameters	90.07±.001%
CNN with KerasTuner hyperparameters	88.31±.001%

Individual embedding-based text classification performance: Table 6.20 illustrates the performance of individual embeddings with different baseline text classification models' performance. *Eighteen* classification models have been produced using the combination of the best-performing three individuals and six classification methods (e.g., SGD, Livsvm, LSTM, GRU, CNN & VDCNN). The four corpora produce 72 classification models (e.g., 18 for BARD, 18 for IndicNLP, 18 for Prothom-Alo & 18 for BTCC11). Table 6.20, *shows only the top-performed classifiers with four corpora*

The BARD corpus maximum accuracy of 94.17±.001 from the GloVe+VDCNN method due to the small number of categories (e.g., 5) and a large number of the training data sample (e.g., 263,299), including distinguished data distribution. The BARD corpus minimum performance was obtained from the GloVe+SGD method owing to the curse of large feature dimensions and insufficient trainable parameters found in SGD classification methods. The IndicNLP corpus has achieved a maximum accuracy of 85.57±0.001. Due to a shortage of generalized feature distribution and a lack of training samples, the IndicNLP corpus accuracy is lower than others. The Prothom-Alo corpus obtained 90.69±0.001 accuracy and compared to better than others. The BTCC11 corpus maximum accuracy

Table 6.20: Test phase statistical performance of individual embeddings with different classifiers, Corpus, testing texts (Ts)

Corpus	#Class(#Ts)	Models	A(%)	MA (%)			WA (%)		
				P	R	$F1$	P	R	$F1$
BARD	5 (112,921)	GloVe+SGD	75.12±.001	72	70	71	74	75	75
		GloVe+LSTM	90.08±.001	88	87	88	90	89	90
		GloVe+VD-CNN	94.17±.001	92	91	92	94	94	94
Indic-NLP	6 (2,826)	GloVe+LSTM	81.92±.001	78	79	78	81	82	81
		GloVe+VD-CNN	85.57±.001	82	82	81	86	85	85
Prothom-Alo	8 (63,888)	GloVe+LSTM	87.53±.001	87	86	87	87	87	87
		GloVe+VD-CNN	90.69±.001	90	89	90	91	90	91
BTCC11	11 (80,931)	GloVe+LSTM	82.71±.001	81	82	81	82	82	82
		FastText+CNN	85.03±.001	84	82	83	83	83	83
		Fast-Text+LSTM	81.10±.001	79	79	79	80	81	80
		GloVe+VD-CNN	87.43±.001	84	85	85	87	87	87

of 87.43±.001% using the GloVe+VDCNN. The sequential (e.g., LSTM & GRU) and statistical (e.g., SVD & Livsvm) methods cannot properly capture BTCC11 corpus morphological and semantics features. As a result, their performance is comparatively lower for VDCNN and CNN.

This work uses three individual embedding methods (GloVe, FastText, and Word2Vec) with different hyperparameters combinations (i.e., three embedding dimensions and two contextual windows), which produced 18 individual embeddings. The GloVe model is frequency-based, and Word2Vec and FastText are neural embedding-based. The FastText model can train with the sub-word features and overcome the OOV issues. Based on the intrinsic evaluation, we have selected three individual embeddings (i.e., 1 for each embedding technique). The extrinsic performance is done only for the three best-performed embedding and meta-embeddings. Due to article length issues, The best-performed text classification model's performance. The FastText+CNN obtained 85.03% accuracy on BTCC11, whereas the FastText+LSTM obtained the maximum accuracy of 81.10%. The sequential model (LSTM) cannot perform well due to vanishing gradient problems.

Meta-embedding-based text classification performance Eighteen classification models have been produced using the combination of the best-performing

three Meta-Embedding and six classification methods (e.g., SGD, Livsvm, LSTM, GRU, CNN & VDCNN). The four corpora produce 72 classification models (e.g., 18 for BARD, 18 for IndicNLP, 18 for Prothom-Alo & 18 for BTCC11). Table 6.21, *shows only the top-performed model performance.*

Table 6.21: Test phase statistical performance of meta-embedding with different classifiers, Corpus, testing texts (Ts)

CC	#Class (#Ts)	Models	A(%)	MA (%)			WA (%)		
				P	*R*	*F*1	*P*	*R*	*F*1
BARD	5 (112,921)	AVG-M+VDCNN	94.82±.001	92	90	91	95	94	95
		AVG-M+LSTM	91.92±.001	89	88	89	92	91	92
		Proposed(AVG-M+CNN)	**95.92±.001**	**93**	**92**	**93**	**96**	**96**	**96**
Indic-NLP	6 (2,826)	AVG-M+VDCNN	86.54±.001	82	**83**	**81**	85	86	86
		AVG-M+LSTM	83.49±.001	82	79	81	83	83	82
		Proposed(AVG-M+CNN)	**87.44±.001**	**84**	80	**81**	**87**	**87**	**87**
Prothom-Alo	8 (63,888)	AVG-M+VDCNN	91.51±.001	**91**	90	**91**	92	92	91
		AVG-M+LSTM	90.76±.001	89	90	88	90	89	90
		Proposed(AVG-M+CNN)	**93.10±.001**	**91**	**91**	**91**	**93**	**93**	**93**
BTCC11	11 (80,931)	AVG-M+VDCNN	89.12±.001	86	85	86	88	89	89
		AVG-M+LSTM	85.89±.001	85	84	85	85	85	85
		Proposed(AVG-M+CNN)	**90.07±.001**	**88**	**88**	**88**	**90**	**90**	**90**

The BARD corpus uniformly accomplished a maximum accuracy of 95.92±.001 due to the small number of categories (e.g., 5) and a large number of the training data sample (e.g., 263,299), including distinguished data distribution. The proposed method achieves the IndicNLP corpus maximum accuracy of 87.44±.001. The IndicNLP corpus is also morphologically rich, i.e., Sadhu-bhasha and Cholito-bhasha texts are present. This is why IndicNLP is quite a low-performing corpus compared to the other three corpora.

The maximum accuracy of 93.10±.001 is obtained for the Prothom-Alo corpus. The BTCC11 corpus maximum accuracy of 90.07±.001%, maximum MA of 88% and maximum WA of 90% using the AVG-M+CNN method (e.g., proposed method), whereas minimum accuracy of 82.71±.001% is obtained for GloVe+LSTM method. The BTCC11 corpus error rate is 10% due to numerous categories, multiple domain texts (e.g., Newspaper, Social media & Blogs), morphological variation texts, and semantically intra-related categories (e.g., Crime & law-justice) are also included in this corpus. The sequential (e.g., LSTM & GRU) and statistical (e.g., SVD & Livsvm) methods cannot properly capture BTCC11 corpus

morphological and semantics features. As a result, their performance is comparatively lower for VDCNN and CNN.

Individual vs. Meta-embedding-based Text Classifications: From Tables 6.4 6.2, 6.6 and 6.7, i.e., *intrinsic performance of embeddings* concluded that the average Meta-Embedding (AVG-M) showed the best performance for word-level morphological analysis (e.g., semantic & syntactic) due to captured word-level semantic and syntactic feature from multiple embeddings and overcame the OOV issues. In contrast, from Table 6.20 and Table 6.21, i.e., *extrinsic evaluation performance* presumd that the single multi-kernel CNN model delivered the best sentence and document level performance. This could be due to capturing the local and global text features by the CNN classifier. The second-best classification performance has been obtained by the VDCNN model and is very close to CNN. Finally, The word level, sentence level, and document level semantic and syntactic best performance are achieved by AVG-M Meta-embedding and CNN classification methods, e.g., AVG-M+CNN for four classification corpora as well as four intrinsic evaluation datasets (e.g., Semantic, Syntactic, Relatedness & analogy tasks dataset).

Impact of K-fold cross-validation on AVG-M+CNN accuracy: The k-fold cross-validation is evaluated sequentially for the whole dataset, which is essential for small-size datasets [155]. However, in this research, we investigated the performance of k-fold cross-validation on the developed corpus (i.e., BTCC11). We investigated the three-fold and five-fold cross-validation. The three-fold cross-validation has obtained an average accuracy of 89.64%, a precision of 88.21%, a recall value of 89.98%, and an F1-score of 89.08%. On the other hand, the five-fold cross-validation achieved an average accuracy of 90.63%, a precision of 89.36%, and a recall value of 90.02%. The five-fold cross-validation obtained 0.56% higher accuracy than the three-fold cross-validation, whereas the five-fold cross-validation achieved a 0.43% lower accuracy than the proposed AVG-M+CNN system (90.07%). This difference could have happened due to the variation in validation and training sample size.

Impact of Out-of-vocabulary (OOV) on CNN accuracy: We have evaluated the impact of OOV tokens. Table 6.22 shows the impact of OOV tokens on CNN accuracy regarding the BTCC11 corpus. BTCC11 corpus consists of a total of 656,274 unique words in the training set and a total of 414,878 unique words in the testing set. GloVe and Word2Vec embedding techniques offered an

OOV of 13,125 (for the training set) and 7,467 (for the testing set). In contrast, the FastText sub-word model is a zero OOV word. FastText sub-word model segmented a word as 3-gram tokens and produced the word feature using the subsumed-up segmented tokens. The individual embedding with CNN (GloVe+CNN) obtained a maximum accuracy of 86.37%, whereas the proposed average meta-embedding with CNN (AVG-M+CNN) achieved a maximum accuracy of 90.07%. The individual GloVe embedding captures the local and global word semantics but does not handle the OOV words. In contrast, FastText sub-word embedding correctly captures the syntactic features and handles OOV issues by sub-token features. The best-performed GloVe and FastText sub-word models combination produced three meta-embedding (i.e., CONC-M, CONC-SVD-M and AVG-M). AVG-M with CNN achieved the maximum accuracy of 90.07% due to better handling of the OOV issues and capturing both semantic and syntactic features.

Table 6.22: Summary of OOV impact on CNN accuracy

Model	OOV in BTCC11 (Train/Test)	A(%)
GloVe+CNN	13125/7467	86.37±.001
FastText+CNN	00/00	85.03±.001
Word2Vec+CNN	13125/7467	84.9±.001
CONC-M+CNN	00/00	88.78±.001
CONC-SVD-M+CNN	00/00	87.85±.001
Proposed(AVG-M+CNN)	**00/00**	**90.07±.001**

The individual GloVe embedding captures the local and global word semantics but does not handle the OOV words. In contrast, FastText sub-word embedding correctly captures the syntactic features and handles OOV issues by sub-token features. Finally, the best-performing GloVe and FastText sub-word models combined form the three meta-embedding, i.e., CONC-M, CONC-SVD-M, and AVG-M. AVG-M with CNN achieved the maximum accuracy of 90.07% due to handling the OOV issues and capturing both semantic and syntactic features.

AVG-M+CNN Accuracy on Other Languages: To investigate the effectiveness of the proposed system (AVG-M+CNN), we have used a Covid-19 text classification corpus in English [156]. The domain-specific GloVe, FastText, and Word2Vec with CNN achieved an accuracy of 88.89%, 87.32%, and 82.68%, respectively. In contrast, the meta-embedding (for GloVe & FastText) with CNN (i.e., AVG-M +CNN) obtained the highest accuracy of 92.46% due to capturing the holistic feature and improving the local semantic feature representations.

Impact of Multilingual transformer-based Models: Twelve classification models have been produced using the four corpus and four transformer-based frameworks. The four transformer models (e.g., M-BERT, Indic-BERT, MuRIL & M-Distil-BERT) and proposed model (e.g., AVG-M+CNN) performance are shown in Table 6.39. Accordingly, the proposed system accomplished the maximum accuracies of 95.92±.001, 87.44±.001 93.10±.001 and 90.07±.001% respectively for BARD, IndicNLP, Prothom-Alo, and BTCC11 corpus. Whereas the minimum 93.86±.001% accuracy for BARD corpus using the M-BERT method and the minimum 81.85±.001% accuracy for IndicNLP corpus using the Indic-BERT method. The four transformer models obtain 91.78±.001% minimum accuracies for the Prothom-Alo corpus. The M-Distil-BERT and proposed model has achieved the maximum performance (e.g., 90.07±.001% accuracy), and another two models have attained 88.69±.001% minimum accuracy for BTCC11 corpus.

Table 6.23: Text classification test phase statistical performance of transformers-based and proposed models

Corpus	#Class (#Ts)	Models	A(%)	MA (%)			WA (%)		
				P	R	$F1$	P	R	$F1$
BARD	5 (112,921)	M-BERT	93.86±.001	90	91	90	93	94	93
		Indic-BERT	94.95±.001	92	91	92	95	94	95
		M-Distil-BERT	94.97±.001	92	**92**	92	95	95	95
		Proposed(AVG-M+CNN)	**95.92±.001**	**93**	**92**	**93**	**96**	**96**	**96**
Indic-NLP	6 (2,826)	M-BERT	86.04±.001	84	84	84	87	86	86
		Indic-BERT	81.85±.001	74	74	74	83	82	82
		M-Distil-BERT	86.83±.001	83	**83**	**83**	**88**	**87**	**87**
		Proposed(AVG-M+CNN)	**87.44±.001**	**84**	80	81	87	**87**	**87**
Prothom-Alo	8 (63,888)	M-BERT	91.78±.001	89	90	90	91	91	91
		Indic-BERT	91.88±.001	90	90	90	92	92	92
		M-Distil-BERT	92.79±.001	**91**	**91**	**91**	92	92	92
		Proposed(AVG-M+CNN)	**93.10±.001**	**91**	**91**	**91**	**93**	**93**	**93**
BTCC11	11 (80,931)	M-BERT	88.69±.001	88	87	88	89	89	88
		Bangla-Electra	89.13±.001	89	88	88	89	89	89
		Indic-BERT	88.90±.001	88	88	88	89	89	89
		M-Distil-BERT	89.98±.001	89	89	88	89	89	89
		MuRIL	**90.05±.001**	88	87	87	89	89	89
		Proposed(AVG-M+CNN)	**90.07±.001**	88	88	**88**	**90**	**90**	**90**

Transformer-based frameworks have achieved state-of-the-art performance for English and some other morphologically ideal languages [157]. Bengali and some other low-resource languages, such as Italian, cannot perform better using a multi-lingual transformer-based model [2]. The multi-lingual transformer-based language model text classification tasks performance for low resource languages is not better because of *fewer number vocabularies*, *tokenization operation* and *morphologically ill-matched*.

Recently, MuRIL has achieved SOTA performance for most of the significant Indic text language processing tasks. We evaluated the MuRIL fine-tuning performance using the developed corpus (i.e., BTCC11). The MuRIL obtained an average accuracy of 90.05% with an overall training time of 21 hours and 45 minutes. Although the MuRIL accuracy is 0.02% lower than the proposed AVG-M+CNN model (i.e., 90.07%), it gained maximum performance compared to other baseline models (i.e., Indic-BERT, M-Distil-BERT, and M-BERT).

The multi-lingual transformer-based model works with more than one hundred languages, for a particular language has minimal vocabulary sets. In this scenario, 95% of Bengali words are directly not found in the transformer-based models (e.g., M-BERT, Indic-BERT & M-Distil-BERT) [25]. The *tokenization operation* is applied to the transformer-based frameworks when the exact word is not found in the vocal list. In this case, the tokenization operation has broken the word and tries to match the broken word token with the vocab list. This process continues until a single token matches the vocab list. So the tokenization operation changes a word's semantic and syntactic meaning, and most of the low-resource languages lost their text semantics due to tokenization operation and ill-matched techniques [158].

The proposed system trained with the most extensive vocabulary sets (e.g., 2,867,701) directly mapped with an embedding model and carried out the best semantic and syntactic meaning-based feature vector. As a result, 98.76% of words are directly matched with our developed Meta-embedding model. Therefore, the classification models can produce sentence-level and document-level semantically and syntactically correlated models. These models can achieve better classification accuracy. Thus, the transformer-based models have not achieved the best result compared to the proposed method due to the shortage of vocabulary and lack of morphological variant features. The transformer-based models cannot be performed well in other languages for the same reasons, such as Italian.

Comparison with Past Bengali Text Classification Models Few studies have been conducted on text classification in Bengali in recent years [102,

159, 160, 2]. Most of these studies used machine learning and deep-learning techniques evaluated on the developed datasets [159, 149, 2], whereas few works evaluated using the publicly available datasets [102, 160]. Due to the unavailability of other research datasets, contemporary techniques were applied to the four different corpora for comparison. Table 6.24 summarizes the comparison between the proposed (AVG-M+CNN) and past text classification techniques in Bengali. The black underlined score indicates the maximum accuracy for other methods. The blue and red colors indicate training time (e.g., convergence time with corresponding epoch numbers).

The comparative analysis revealed that the proposed system (e.g., AVG-M+CNN) achieved the highest accuracy of 95.92±.001 (BARD), 87.44±.001 (IndicNLP), 93.10±.001 (Prothom-Alo) and 90.07±.001% (BTCC11) for four datasets, respectively. Whereas the minimum accuracy of 90.07±.001% (BARD), 82% (IndicNLP) and 87% (Prothom-Alo) are obtained using Char-Embedding+CNN technique [160]. The M-BERT required the highest training time (e.g., training convergence time) of 29 hours and 56 minutes [102], whereas the AVG-M+CNN consumed a minimum training time (23 minutes).

We have also investigated the state-of-the-art Indic transformer-based language model (MuRIL) on BTCC11. The results revealed that the proposed system (AVG-M+CNN) achieved a higher accuracy of 90.07% than the other multilingual transformer-based language models (M-DistilBERT (89.98%), IndicBERT (88.90%), M-BERT (88.69%) and MuRIL (90.05%). The IndicBERT and MuRIL are both Indic language-specific language models, and MuRIL achieved the best accuracy among the baseline SOTA approaches. However, the proposed mete-embedding gained maximum accuracy (0.02% higher than the MuRIL) due to capturing more semantic and syntactic feature extraction. In comparison, the proposed method attained a better performance concerning GloVe+VDCNN because of the combined different embedding features (e.g., ME) and tuned the CNN hyperparameters using an automatic process (e.g., Algorithm 9).

6.5.4 Performance of Attention-based Meta-Embedding with ACNN Model

The six deep learning baseline methods, the proposed CovTiNet method, and the four transformer-based fine-tuning methods produced 40 models (where deep learning + CovTiNet contributed 36 models and the transformer-based technique contained four models). Among these 40 models, Table 6.25 shows the performance of 17 models (six best-performed models, six worst-performed, and four

Table 6.24: Comparison summary among proposed and previous research concerning accuracy (A), convergence time (T in hrs), corpus size (CS), and epoch numbers (E_p).

Corpus	CS(MB)	Methods	A(%)	T/E_p
BARD	1594.10	**Proposed(AVG-M+CNN)**	**95.92±.001**	7:51/150
		GloVe+VDCNN	94.17±.001	19:14/180
		M-BERT	93.86±.001	29:56/10
		Wor2Vec+NN	91.89±.001	6:20/160
		Chr-Em+CNN	90.03±.001	6:46/151
IndicNLP	60.00	**Proposed(AVG-M+CNN)**	**87.44±.001**	0:23/17
		GloVe+VDCNN	85.57±.001	0:38/19
		M-BERT	86.04±.001	1:15/10
		Wor2Vec+NN	83.10±.001	0:28/19
		Chr-Em+CNN	81.68±.001	0:26/18
Prothom-Alo	1095.40	**Proposed(AVG-M+CNN)**	**93.10±.001**	4:20/170
		GloVe+VDCNN	90.69±.001	10:41/150
		M-BERT	91.78±.001	16:16/10
		Wor2Vec+NN	88.06±.001	3:07/190
		Chr-Em+CNN	87.71±.001	3:17/155
BTCC11	1212.4	**Proposed(AVG-M+CNN)**	**90.07±.001**	6:30/160
		MuRIL	**90.05±.001**	21:29/10
		GloVe+VDCNN	87.43±.001	15:17/200
		M-BERT	88.69±.001	20:29/10
		Wor2Vec+NN	83.10±.001	4:12/200
		Chr-Em+CNN	86.73±.001	4:20/170

transformer-based fine-tuned models), including the proposed CovTiNet for CTI tasks. The extrinsic evaluation reported the CTI task performance based on the learning ability and intelligence of the model.

Results revealed that the proposed model (CovTiNet) achieved the maximum accuracy of $96.61 \pm 0.001\%$, whereas GloVe+LibSVM achieved the minimum accuracy ($82.26 \pm 0.001\%$). The proposed attention-based fusion and position encoding improved the accuracy of $14.35 \pm 0.001\%$ compared to GloVe+LibSVM, $5.72 \pm 0.001\%$ from GloVe+LSTM and $4.92 \pm 0.001\%$ from CNN. There are two critical reasons for improving the proposed CovTiNet performance compared to other models: (i) the proposed position encoding extracts the word-level syntactic information, and (ii) the attention-based fusion enhances the quality of the semantic features representation. Thus, the combined attention and position

Table 6.25: CTI task performance of the proposed (CovTiNet) and baseline models. The MA and WA values are rounded up to two decimal points

Models	$A(\%)$	MA (%)			WA (%)		
		P	R	F_1	P	R	F_1
GloVe+LibSVM	82.26±.001	82	82	82	82	82	82
GloVe+LSTM	90.89±.001	91	91	91	91	91	91
GloVe+BiLSTM	92.54±.001	93	93	93	93	93	93
GloVe+VDCNN	93.17±.001	93	93	93	93	93	93
GloVe+DCNN	92.32±.001	92	92	92	92	92	92
GloVe+CNN	91.69±.001	92	92	92	92	92	92
APeAGF+LibSVM	84.75±.001	85	85	85	85	85	85
APeAGF+LSTM	92.64±.001	93	93	93	93	93	93
APeAGF+BiLSTM	95.14±.001	95	95	95	95	95	95
APeAGF+VDCNN	93.65±.001	92	91	92	94	94	94
APeAGF+DCNN	92.97±.001	93	93	93	93	93	93
APeAGF+CNN	94.13±.001	94	94	94	94	94	94
BERT-M	95.88±.001	96	96	96	96	96	96
DistilBERT-M	95.01±.001	95	95	95	95	95	95
IndicBERT	93.13±.001	93	93	93	93	93	93
ELECTRA-Bengali	96.19±.001	96	96	96	96	96	96
CovTiNet (Proposed)	**96.61±.001**	**97**	**97**	**97**	**97**	**97**	**97**

encoding improves linguistic understanding concerning Bengali. In contrast, the statistical classifier (e.g., LibSVM), the sequential classifier (e.g., LSTM), and the Convolutional classifier (e.g., CNN) with non-contextual embedding (e.g., GloVe, FastText, and Word2Vec) can not adequately represent the Bengali textual features based on semantic and syntactic meaning.

Comparison with Previous Research According to this work exploration, no significant research has been done to identify or classify COVID-19 text in Bengali, including corpus development. Thus, this study embraced several contemporary methods that have been examined on similar tasks in other language datasets. For consistency, a few past techniques [161, 162, 163, 164, 2, 165, 166] have been implemented on the developed dataset (i.e., $BCovC$) and compared their performance with the proposed approach (CovTiNet). Table 6.26 shows the comparison among various techniques in terms of accuracy (A_c), training time in hours (TTH), and GPU memory consumption in GB (GMCG) to perform CTI tasks.

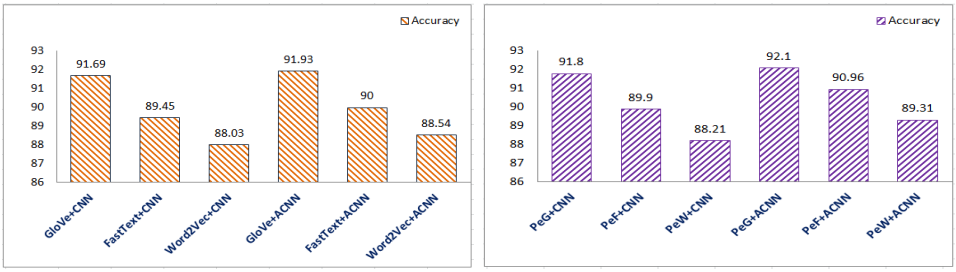
The transformer-based fine-tuned models (BERT-M, IndicBERT, and DistilBERT) consumed too much GPU memory and training time compared to CovTiNet. However, their accuracy is significantly lower than the CovTiNet.

Table 6.26: Comparison between the proposed and recent techniques in terms of A_c, TTH and $GMCG$ on $BCovC$

Methods	$A(\%)$	TTH	GMCG
BiLSTM+FastText [161]	91.47	0.53	6.5
CNN+FastText [162]	89.45	**0.43**	**3.8**
VDCNN+Word2Vec [163]	90.68	0.62	5.6
ELECTRA-Bengali [164]	96.19	0.68	6.2
BERT-M [2]	95.88	3.03	7.9
DistilBERT [165]	94.88	0.70	6.01
IndicBERT [166]	93.13	2.33	7.6
CovTiNet (Proposed method)	**96.61**	0.51	4.5

Because of the smaller vocabularies in the language model and significant morphological variation (semantic and syntactic) of the Bengali language, the transformer-based model showed inferior performance. The ELECTRA-Bengali is a monolingual language model whose accuracy (96.19%) is much better than the BERT-M (95.88%), IndicBERT (93.13%), and DistilBERT (94.88%) due to monolingual effect due to the single language model gained much attention for semantic and syntactic representations than multilingual models [32].

This section demonstrates how the CovTiNet gained better performance than other models due to incorporating attention-based positional embedding feature fusion and attention operation on the CNN method. Fig. 6.3a illustrates the impact of attention-based CNN (ACNN) embedding on the single embedding models (e.g., GloVe, FastText, and Word2Vec).



(a) B_a+(CNN,ACNN) **(b)** (B_a, P_e)+(CNN,ACNN)

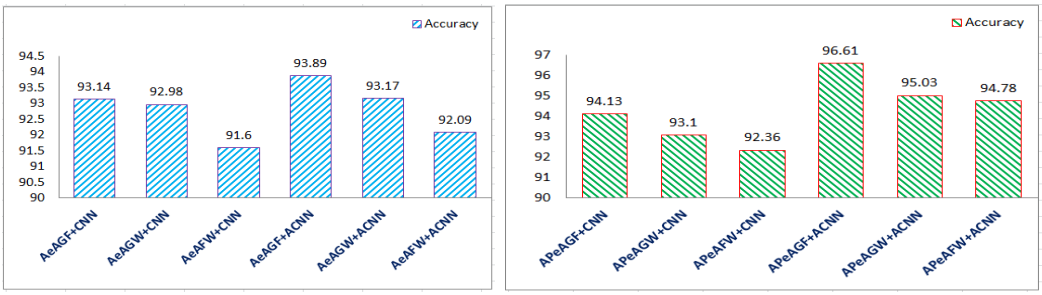
Figure 6.3: Impact of position encoding (P_e) on embedding models for CTI task performance with CNN and ACNN

Due to attention operation on the CNN method, the document-level semantic and syntactic feature extraction has an accuracy improvement of about 0.55% by FastText+CNN (from 89.45% to 90.00%). Fig. 6.3b depicts the impact of position encoding operation with the three single embedding models: position encoding with GloVe (PeG), position encoding with FastText (PeF), and position encoding with Word2Vec (PeW). Figure 6.3b illustrated that the combi-

nation of position encoding on embedding models and attention operation on CNN achieved a notable performance improvement in the CTI tasks. The position encoding and attention operation have improved by about 0.17% accuracy of GloVe (i.e., 91.93% for GloVe+ACNN, 92.1% for PeG+ACNN), 0.96% accuracy improvement of FastText (i.e., 90.00% for FastText+ACNN, 90.96% for PeF+ACNN) and 0.77% improvement achieved for Word2Vec embedding (i.e., 88.54% for Word2Vec+ACNN, 89.31% for PeW + ACNN). Fig. 6.3 depicts the overall performance of ACNN and the position encoding with embedding models, which are better than CNN with single embedding models.

The intrinsic evaluation results (in Sec. 6.4 showed enhanced performance on CTI tasks due to the *attention-based average feature fusion*. Therefore, we analyzed the impact of attention-based average feature fusion and position encoding operation on CNN and ACNN on CTI (Fig. 6.4a). In particular, we investigate three operations: (i) attention-based average feature fusion of GloVe+FastText (APeAGF), (ii) attention-based average feature fusion of GloVe+Word2Vec (AeAGW), and (iii) attention-based average feature fusion of FastText+Word2Vec (AeAFW).

It is revealed that the attention-based average feature fusion (APeAGF+ACNN) has enhanced the maximum accuracy of 0.75% compared to APeAGF+CNN (Fig. 6.4a). Fig. 6.4b shows the attention-based position encoding average feature fusion GloVe+FastText (APeAGF) and Attention operation on CNN (ACNN). The CovTiNet system achieved the best accuracy of 96.61%. Regarding attention operation on CNN, the maximum accuracy of 2.42% is improved compared to APeAGF+CNN (94.13%). Thus, it is confirmed that the attention-based position encoding average feature fusion and attention operation on CNN have a significant performance improvement in performing CTI tasks in Bengali.



(a) attention-based average feature fusions (APeAGF, AeAGW, AeAFW)

(b) positional embedding average feature fusions (APeAGF, APeAGW, APeAFW)

Figure 6.4: Impact of attention-based and positional embedding-based average feature fusions on CTI task performance with CNN and ACNN

Figs. 6.3 and 6.4 showed that the attention-based position encoding feature

fusion is better than the single embeddings. The attention operation on CNN has significantly improved the semantic and syntactic features representation at sentence and paragraph levels, whereas the position encoding operation improved the contextual features representation. Therefore, the combination of attention, feature fusion, and position encoding showed enhanced CTI task performance by CovTiNet.

Ablation Studies Evaluations In the text-to-feature extraction module, the three best-performed non-contextual embedding methods, i.e. Word2Vec, GloVe, and FastText, as well as the three best-performed attention-based feature fusion embeddings (i.e., APeCGF, APeCpGF, and APeAGF) are used for Bengali text-to-feature extraction purposes. However, the word-level performance analysis (i.e., intrinsic evaluators) is summarized in Table 6 and Table 7. These results drastically drop the single embedding performance compared to the attention-based feature fusion performance. For example, the best performing attention-based averaging of GloVe and FastText-based features fusion (i.e., APeAGF) improved the Spearman correlation of 11.91%, 18.46%, and 2.23% for single embedding FastText, Word2Vec, and GloVe respectively for Semantic similarity dataset (i.e., S_m). Similarly, the syntactic, relatedness and analogy task dataset performs better using APeAGF embedding than other embeddings. From this ablation analysis, the text-to-features extraction module removed the single embedding methods (i.e., GloVe, FastText & Word2Vec) and removed the other two attention-based feature fusion embeddings (i.e., APeCGF and APeCpGF). The position-based information significantly impacts text identification performance, as depicted in Figure 6. This study included the position information with APeAGF and named an attention-based position embedding averaging of GloVe and FastText (APeAGF). Finally, the APeAGF is selected for the part of the CovTiNet module (Figure 5).

In the Covid text identification module, the ablation analysis initially considers six deep learning methods (i.e., CNN, VDCNN, DCNN, ACNN, LSTM, and BiLSTM) and four transformer-based language model fine-tuning methods (i.e., BERT-M, DistilBERT-M, ELECTRA-Bengali and IndicBERT). Among these ten methods, the ablation analysis carried the attention-based CNN (i.e., ACNN) achieved a better performance in terms of accuracy in the Bengali Covid text corpus (i.e., BCovC). The ten text identification methods' performance is summarized in Table 8, where Covid text identification performance is evaluated using the different combinations of single embeddings and attention-based feature fusion embeddings with ten text identification methods. So, the ablation

analysis concludes the proposed CovTiNet, i.e., a combination of attention-based position embedding averaging of GloVe and FastText (APeAGF) and attention-based CNN (ACNN) achieved the best performance in BCovC text identification corpus and word level intrinsic evaluation dataset (i.e., IEDs)

6.5.5 Extrinsic Performance of Transformer-based Models

In this section, we examine the performance of transformer-based language models and outline the limitations they face in the context of resource-constrained text classification. Subsequently, we provide an overview of the performance of the proposed attention-based early fusion model in the domains of Bengali document classification and authorship attribution. The following subsections delve into the details of these model performances.

6.5.5.1 Performance of Arabic Covid Text Classification

This research analyzes the performance of baselines and the proposed models in various aspects (*Answer to the fourth research question [**RQ4**]*). The ablation analysis illustrates how to carry out the best-performing task-specific system built from a set of baseline methods. The impact of language-specific transformer-based language models, embedding model's vocab size, and hyperparameters of transformer models has also been demonstrated.

Outcomes of Baseline Models Table 6.27 shows the performance of 18 baseline models (06 transformer-based, 12 DL-based).

Table 6.27 revealed that the maximum accuracy of 98.52±.001 achieved from the mono-lingual AraBERT models. This outcome could be due to the ability of the mono-lingual transformer-based embedding to capture the word-level semantic and syntactic features. In contrast, the AraBERT also captures local and global Arabic text document semantics and overcomes the out-of-vocabulary (OOV). Among 18 baseline methods, Arabic Word2Vec embedding ($AE_i^N \in \{Word2Vec\}$) with VDCNN obtained the minimum accuracy of 86.77±.001, due to the inability of $AE_i^N \in \{Word2Vec\}$ embedding to capture the lack of word-level semantic and syntactic features. In addition to the inability of VDCNN to capture sentence and document-level semantics. Based on the baseline method's performance, this study selected the four best-performing models for the late fusion purpose (i.e., B_{m1}, B_{m2}, B_{m3}, and B_{m4}).

Table 6.27: Performance of Arabic CovTI baseline methods where the bold indicates the score of top-4 models

Models	A (%)	P (%) CO/NC	R (%) CO/NC	MA (%) P R $F1$	WA (%) P R $F1$
AraBERT	**98.52±.001**	**97/99**	**99/98**	**98 99 99**	**99 99 99**
BERT-Arabic	**97.43±.001**	**95/99**	**99/96**	**97 98 97**	**98 97 97**
mBERT	95.46±.001	92/99	98/93	95 96 95	96 95 95
XML-RoBERTa	**96.32±.001**	**93/99**	**99/95**	**96 97 96**	**96 96 96**
mDeBERTa-V3	**96.35±.001**	**93/99**	**99/94**	**96 97 96**	**97 96 96**
mDistilBERT	93.97±.001	89/99	99/90	94 95 94	94 94 94
GloVe+LSTM	91.16±.001	85/98	97/84	90 91 91	91 91 91
GloVe+BiLSTM	92.08±.001	86/98	98/87	92 93 92	92 93 92
GloVe+CNN	90.03±.001	83/97	98/92	89 90 90	90 91 90
GloVe+VDCNN	88.27±.001	88/96	97/87	87 88 88	88 88 88
Word2Vec+LSTM	90.50±.001	82/97	96/83	91 90 90	90 91 90
Word2Vec+BiLSTM	91.68±.001	87/97	97/86	93 92 92	92 93 92
Word2Vec+CNN	88.52±.001	85/97	97/96	89 88 88	88 88 88
Word2Vec+VDCNN	86.77±.001	83/95	96/82	86 85 85	86 85 95
FastText+LSTM	90.91±.001	85/97	96/86	92 91 91	92 92 92
FastText+BiLSTM	91.86±.001	88/98	97/89	92 92 92	92 92 92
FastText+CNN	89.82±.001	84/96	97/83	90 89 89	90 90 90
FastText+VDCNN	87.69±.001	84/96	97/83	88 87 87	88 88 88

Impact of Late Fusion on Accuracy Table 6.28 shows the performance of four best-performed baseline models with late fusion (i.e., B_{m1} : AraBERT, B_{m2} : BERT-Arabic, B_{m3} : mDeBERTa-V3, and B_{m4} : XML-RoBERTa). V_A and F1-V_A denote the voting-based late fusion accuracy and F1-score, whereas W_A and F1-W_A subsequently denote the weighted average-based late fusion accuracy and F1-score.

Table 6.28: Performance of the fusion-based four best-performed models where the top model's score marked as bold

Late Fusion Model	V_A (%)	W_A (%)	F1-V_A(%)	F1-W_A(%)
(B_{m1}, B_{m2}, B_{m3}, B_{m4})	98.46±.001	98.49±.001	98.35	98.38
(B_{m1}, B_{m2}, B_{m3})	**98.75±.001**	**98.89±.001**	**98.65**	**98.81**
(B_{m1}, B_{m2}, B_{m4})	98.66±.001	98.70±.001	98.54	98.59
(B_{m2}, B_{m3}, B_{m4})	98.54±.001	98.61±.001	98.43	98.50

Table 6.28 indicates that the minimum weighted average and voting-based late fusion accuracy are 98.46±.001 and 98.49±.001 obtained from the first best model combination (i.e., (B_{m1}, B_{m2}, B_{m3}, B_{m4})). This outcome could have happened due to the inability of the fourth model (i.e., B_{m4}) to mispredict some samples. The second combination (i.e., (B_{m1}, B_{m2}, B_{m3})) achieved the best accuracy of

98.75±.001 (for voting-based late fusion) and 98.89±.001 (for weighted average-based late fusion). This performance could be due to the ability of the three best-preformed models' strengths to gain maximum soft-max scores. In comparison among four late fusion combinations for two late fusion methods (i.e., voting & weighted average), the weighted average-based late fusion has achieved the best performance for the second combination of three best-performed models (i.e., B_{m1}, B_{m2} & B_{m3}). *Thus, the weighted average-based late fusion with the second combination (i.e., (B_{m1}, B_{m2}, B_{m3})) has been selected as the proposed system* (i.e., called AraCovTexFinder).

Impact of Hyperparameters on Fine-tuned Accuracy The performance of the fine-tuned transformer-based language model largely depends on their hyperparameters [142]. Based on the GPU and time budgets, this research tunes the three most influential hyperparameters, such as batch size (B_S), maximum sequence length (M_{SL}), and the number of epoch (epoch). Due to the time budget, the hyperparameters are fine-tuned on GTX 1070 Ti 8G GPU in two rounds using mono-lingual and multi-lingual language models (i.e., AraBERT & mDeBERTa-V2). The first round uses 1 to 10 epochs. At the same time, the second round network weights are initialized by round one fine-tuned weights and training up to 20 epochs (i.e., 11-20). Table 6.29 shows the two rounds' best-performed accuracy and their hyperparameters configurations.

Table 6.29 revealed that the first round mono-lingual (i.e., AraBERT) best accuracy of 98.53% obtained for the $< B_S : 10, M_{SL} : 100, epoch : 10 >$ configuration. Whereas the round one multi-lingual mDeBERTAa-V3 obtained the maximum accuracy of 96.35 for this configuration, i.e., $< B_S : 12, M_{SL} : 100, epoch : 10 >$. In the second round maximum AraBERT accuracy of 98.47% for the $< B_S : 10, M_{SL} : 100, epoch : 11 >$ configuration and mDeBERTa-V3 accuracy of 96.02% for the $< B_S : 12, M_{SL} : 100, epoch : 11 >$ configuration.

Table 6.29 illustrated that the AraBERT and mDeBERTa-V3 have fallen into the *model overfitting issues* after epochs 11 to upwards due to a shortage of training samples and lack of dataset diversity. Due to morphological variety, AraBERT's best document-level local and global semantics are captured at batch size 10. In contrast, the mDeBERTa-V3 captures at batch size 12, i.e., *takes more samples for better gradients calculation.* In contrast, this study could identify the best possible fine-tuned hyperparameter value for the AraCoV corpus.

Impact of Vocab-size and Language-specificity on Fine-tuned Accuracy: Increasing the vocabulary size of a language model can improve its fine-

Table 6.29: Summary of hyperparameters and their impact on fine-tuned accuracy, B_S for batch size & M_{SL} for maximum sequence length (best combination marked as bold)

Round	Model	Configuration	A(%)
1	AraBERT	{B_S: 8, M_{SL}: 200, epoch: 10}	97.82
		{B_S: 5, M_{SL}: 300, epoch: 10}	96.90
		{B_S: 10, M_{SL}: 100, epoch: 10}	**98.52**
		{B_S: 8, M_{SL}: 150, epoch: 10}	98.07
		{B_S: 10, M_{SL}: 70, epoch: 10}	97.13
	mDeBERTa-V3	{B_S: 9, M_{SL}: 200, epoch: 10}	95.83
		{B_S: 5, M_{SL}: 300, epoch: 9}	94.04
		{B_S: 12, M_{SL}: 100, epoch: 10}	**96.35**
		{B_S: 8, M_{SL}: 150, epoch: 10}	95.91
		{B_S: 12, M_{SL}: 70, epoch: 10}	94.41
2	AraBERT	{B_S: 8, M_{SL}: 200, epoch: 12}	97.63
		{B_S: 5, M_{SL}: 300, epoch: 13}	96.40
		{B_S: 10, M_{SL}: 100, epoch: 11}	**98.47**
		{B_S: 8, M_{SL}: 150, epoch: 12}	97.89
		{B_S: 10, M_{SL}: 70, epoch: 17}	97.05
	mDeBERTa-V3	{B_S: 9, M_{SL}: 200, epoch: 14}	94.90
		{B_S: 5, M_{SL}: 300, epoch: 12}	93.76
		{B_S: 12, M_{SL}: 100, epoch: 11}	**96.02**
		{B_S: 8, M_{SL}: 150, epoch: 15}	93.92
		{B_S: 12, M_{SL}: 70, epoch: 17}	93.89

tuned accuracy, as it allows the model to understand a broader range of words and phrases. However, it can also increase the model's computational requirements and memory usage. This study investigated the fine-tuned performance of transformer-based language models using mono-lingual and multi-lingual models. Specifically, two mono-lingual BERT-based models (AraBERT and BERT-Arabic) and four multi-lingual models (mBERT, mDeBERTa-V2, mDistilBERT, and XLM-RoBERTa-based) were used. As presented in Fig. 6.5, the results indicate that the mono-lingual models achieved higher accuracy and better context-aware semantics for Arabic text than the multi-lingual models. Despite having the most extensive vocabulary size (251,00), the mDeBERTa-V3 model, which supports 100 languages, had lower accuracy than the BERT-Arabic model, which only supports Arabic and has a vocabulary size of 32,000. The highest accuracy of 98.52% is achieved by the mono-lingual AraBERT model, which has a vocabulary size of 64,000. Thus, the results suggest that the fine-tuned performance of transformer-based language models depends on vocabulary size and whether the model is mono-lingual or multi-lingual.

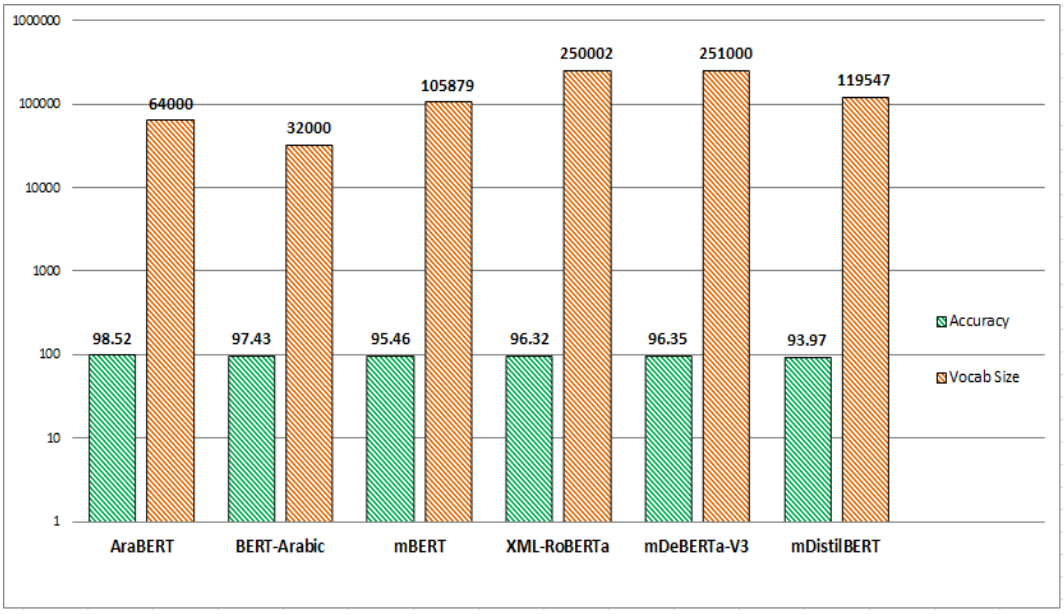


Figure 6.5: Impact of Vocab-size and Language-specificity on Accuracy

Impact of Out-of-vocabulary on Accuracy: This study aimed to examine the impact of out-of-vocabulary (OOV) and character or sub-word-level match (CSM) words on the performance of AraCoVTexFinder. An OOV issue refers to a word not found in the embedding model. In such cases, we use sub-word tokenization, allowing the model to represent OOV words by breaking them into smaller units in the vocabulary. These smaller unit features are then summed up to form the OOV word features (i.e., CSM). Two non-contextual embeddings (GloVe and Word2Vec) could not manage OOV issues [3]. However, using character or subword-level features, the non-contextual FastText and six context-aware language models overcame OOV issues. However, using subword or character-level word features resulted in losing some semantic meaning, which decreased overall system accuracy. Table 6.30 illustrates the classification accuracy of contextual and non-contextual embeddings. The MW denotes the whole match word in the embedding, i.e., not the OOV word. The CSM denotes the character or sub-word level match word (i.e., OOV word).

As demonstrated in Table 6.30, the mono-lingual AraBERT achieved the highest accuracy of 98.52%. This outcome can be attributed to its ability to achieve the maximum direct whole word match (MW) and effectively capture word, sentence, and document-level semantics. In contrast, non-contextual embeddings (i.e., GloVe and Word2Vec) achieved a high MW. However, their accuracy was lower due to the absence of sentence and document-level semantic and syntactic features.

Table 6.30: Summary of OOV impact on accuracy where the top-4 scores marked as bold, CSM: character or subword-level match, OOV: out-of-vocabulary

Model	MW	CSM	A(%)
AraBERT	**3,940**	**93,824**	**98.52±0.001**
BERT-Arabic	**2,631**	**95,133**	**97.43±0.001**
mBERT	1,050	96,714	95.46±0.001
XML-RoBERTa	**1,109**	**96,655**	**96.32±0.001**
mDeBERTa-V3	**1,210**	**96,554**	**96.35±0.001**
GloVe+BiLSTM	79,964	0	92.08±0.001
Word2Vec+BiLSTM	79,964	0	91.68±0.001
FastText+BiLSTM	79,964	17,800	91.86±0.001

Performance of Proposed CovTI System (AraCovTexFinder): An empirical analysis of the baseline model's performance concluded that the late-fusion-based ensembling of the three best-performed transformer models is selected as the proposed system (i.e., AraCovTexFinder). The proposed system and the best-performed three models (i.e., B_{m1}: AraBERT, B_{m2}: BERT-Arabic, and B_{m3}: mDeBERTa-V3) were evaluated using F1-score, category-based accuracy (C_A), and an average accuracy (A_A). Table 6.31 shows the summary of the results.

Table 6.31: Performance of the Proposed system (AraCovTexFinder) and three best-performed where the top scored model marked as bold)

Model	Category	F1-score (%)	C_A (%)	A_A (%)
B_{m1}:AraBERT	C	98.30	99.27	98.52
	NC	98.71	97.98	
B_{m2}:BERT-Arabic	C	97.06	99.16	97.43
	NC	97.72	96.15	
B_{m3}:mDeBERTa-V3	C	95.87	99.05	96.35
	NC	96.73	94.34	
AraCovTexFinder	**Covid**	**98.65**	**99.27**	**98.89**
	Non-Covid	**98.98**	**98.52**	

The AraCovTexFinder system achieved an F1-score of 98.65% in the Covid category (C) and 98.98% in the Non-Covid category (NC). The AraCovTexFinder achieved a category-based accuracy of 99.27% in C and 98.98% in NC with an average accuracy of 98.89%. The best-performed three models also demonstrated high performance in the F1-score, C_A, and A_A. The B_{m1} model achieved the highest F1-score of 98.71% in the NC, and B_{m2} model achieved the highest C_A of 99.16% in C. Overall, the proposed system and the best-performed three models demonstrated strong performance measures because it combines the strengths

of three individual models and can detect patterns well that are not visible in a single model. Late fusion is also more robust to noise and outliers than individual models.

Impact of K-fold Cross-validation on AraCovTexFinder Accuracy K-fold cross-validation is a well-known technique used to evaluate the accuracy of DL-based models. This study investigated the 3-fold and 5-fold cross-validation impact on AraCovTexc. The 3-fold average accuracy of 98.92% and 5-fold average accuracy of 99.13%. The results showed that the accuracy of AraCovTexFinder was improved by 0.24% when using 5-fold cross-validation compared to when the model was trained and tested on the same data. This result indicates that 5-fold cross-validation is an effective technique for evaluating the accuracy of this type of model. In conclusion, 5-fold cross-validation can improve the accuracy of AraCovTexFinder.

6.5.5.2 Performance of Proposed AFuNeT

Table 6.32 illustrates the performance of 100 baseline models across the five corpora. These models employed 60 DL and 40 transformer-based approaches. All models used the default hyperparameters.

Table 6.32 demonstrates the remarkable performance achieved by transformer-based language models, particularly BanglaBERT. With a maximum accuracy of 92.43±.2% for BTCC11, 82.82±.2% for OSBC, 96.48±.2% for BARD, 89.25±.2% for IndicNLP, and 95.87±.2% for Prothom-Alo, BanglaBERT obtained maximum accuracy due to its utilization of contextual word-level attributes and attention-based features at the sentence and document levels.

Several hyperparameters, i.e., maximum sequence length, batch size, and number of epochs, are fine-tuned for DL and transformer models based on the five corpora. One hundred classifier models are developed by combining deep learning and transformer, where DL models contributed 60 and transformer models donated 40. Table 6.33 illustrates the performance of eight best-performed models among 100 models with tuned hyperparameters.

The results presented in Table 6.33 indicate that fine-tuning the hyperparameters based on Bengali corpora has led to improved performance compared to default hyperparameters. The overall BanglaBERT-based model improvement of 0.86% for BTCC11, 0.69% for OSBC, 0.28% for BARD, 1.38% for IndicNLP and 0.10% for Prothom-Alo. This improvement can be attributed to optimizing batch size about the maximum sequence length, resulting in a more precise calculation of global and local gradient values for the Bengali corpora.

Table 6.32: Accuracy of baseline models on five corpora. The bold values indicate the best-performed models. Adding ±0.2 to all results is a way to express the potential variability or uncertainty in the reported performance metrics.

	BTCC11	OSBC	BARD	IndicNLP	Prothom-Alo
Model	A(%)	A(%)	A(%)	A(%)	A(%)
CNN+GloVe	**87.43±.2**	**77.34±.2**	**94.19±.2**	**86.41±.2**	**92.16±.2**
LSTM+GloVe	82.71±.2	75.17±.2	90.12±.2	83.47±.2	91.00±.2
BiLSTM+GloVe	86.03±.2	76.06±.2	91.15±.2	85.18±.2	91.75±.2
GRU+GloVe	81.03±.2	74.61±.2	89.97±.2	82.06±.2	90.37±.2
CNN+FastText	86.00±.2	75.90±.2	93.01±.2	86.19±.2	91.96±.2
LSTM+FastText	81.86±.2	74.49±.2	89.62±.2	82.61±.2	90.08±.2
BiLSTM+FastText	85.48±.2	75.43±.2	90.34±.2	84.20±.2	91.69±.2
GRU+FastText	80.73±.2	73.50±.2	88.53±.2	81.09±.2	90.02±.2
CNN+Word2Vec	84.98±.2	74.31±.2	91.97±.2	84.53±.2	91.16±.2
LSTM+Word2Vec	80.85±.2	73.68±.2	88.47±.2	80.96±.2	89.48±.2
BiLSTM+Word2Vec	83.79±.2	74.40±.2	89.86±.2	83.07±.2	90.00±.2
GRU+Word2Vec	79.07±.2	72.00±.2	87.46±.2	80.89±.2	88.83±.2
BanglaBERT	**92.43±.2**	**82.82±.2**	**96.48±.2**	**89.25±.2**	**95.87±.2**
BELECTRA	89.18±.2	80.19±.2	95.83±.2	87.63±.2	94.00±.2
MuRIL	90.03±.2	82.25±.2	96.14±.2	88.91±.2	95.26±.2
DeBERTa	89.96±.2	80.06±.2	95.07±.2	87.88±.2	94.14±.2
MBERT	88.90±.2	79.58±.2	93.15±.2	86.78±.2	93.89±.2
XLMRoBerta	89.86±.2	80.39±.2	95.68±.2	87.56±.2	94.08±.2
IndicBERT	88.87±.2	78.72±.2	94.97±.2	86.65±.2	93.18±.2
DistilBERT	89.90±.2	78.03±.2	94.98±.2	86.79±.2	93.20±.2

Comparing Accuracy Improvement Between Focal and Cross-Entropy Loss Functions In response to **RQ5**, this research employed the focal loss technique, which led to a notable enhancement in overall accuracy across all five corpora. In Figure 3.9b, it is evident that the IndicNLP corpus exhibits the most significant class imbalance. Table 6.33 reveals that the highest average accuracy improvement, amounting to 1.39%, was achieved with the IndicNLP corpus when using the focal loss function (L^{Fo}). The second-best average accuracy improvement, at 1.04%, was observed with the BTCC11 dataset, also when employing the L^{Fo} loss function. On the other hand, the Prothom-Alo corpus exhibited the lowest average accuracy improvement of 0.37% with the L^{Fo} loss function, as illustrated in Figure 3.10a, indicating fewer class imbalance issues and sufficient samples per class. Furthermore, the OSBC and BARD corpora demonstrated average accuracy improvements of 0.98% and 0.87%, respectively. In summary, the results suggest that focal loss (L^{Fo}) led to improved accuracy in class-imbalanced corpora compared to cross-entropy loss (L^{Cs}) by emphasizing complex samples within the imbalanced categories [140].

Table 6.33: Performance of the fine-tuned best-performed models with hyper-parameters: focal loss (L^{Fo}) and categorical cross-entropy loss

Model	Loss	BTCC11 A(%)	OSBC A(%)	BARD A(%)	IndicNLP A(%)	Prothom-Alo A(%)
BanglaBERT	L^{Fo}	**93.29±.2**	**84.87±.2**	**97.20±.2**	**93.49±.2**	**96.42±.2**
BELECTRA		92.15±.2	80.42±.2	95.97±.2	88.56±.2	94.43±.2
MuRIL		93.22±.2	84.50±.2	97.11±.2	93.41±.2	96.38±.2
DeBERTa		92.85±.2	83.23±.2	96.48±.2	91.75±.2	95.09±.2
MBERT		89.86±.2	80.78±.2	96.65±.2	88.58±.2	94.69±.2
XLMRoBerta		92.87±.2	83.22±.2	97.07±.2	91.25±.2	95.50±.2
IndicBERT		89.10±.2	80.96±.2	95.62±.2	86.97±.2	93.82±.2
DistilBERT		91.38±.2	79.72±.2	96.31±.2	87.33±.2	93.31±.2
BanglaBERT	L^{Cs}	**92.81±.2**	**83.51±.2**	**96.76±.2**	**90.63±.2**	**95.97±.2**
BELECTRA		91.10±.2	80.23±.2	95.86±.2	87.89±.2	94.20±.2
MuRIL		92.06±.2	83.41±.2	96.89±.2	90.53±.2	95.79±.2
DeBERTa		90.96±.2	81.49±.2	95.34±.2	89.25±.2	94.86±.2
MBERT		89.16±.2	80.12±.2	94.75±.2	87.68±.2	94.13±.2
XLMRoBerta		90.96±.2	81.42±.2	95.79±.2	90.48±.2	95.40±.2
IndicBERT		88.97±.2	79.69±.2	95.03±.2	86.70±.2	93.05±.2
DistilBERT		90.31±.2	79.92±.2	95.06±.2	87.06±.2	93.25±.2

Baseline vs. Fine-Tuned Accuracy Comparison Figure 6.6 visually illustrates the accuracy improvement of best-performing eight baseline models and eight fine-tuned models across the five corpora using group violin plots.

In Figure 6.6, the x-axis corresponds to the names of the corpora, while the y-axis signifies the accuracy values. Each group of violin plots portrays both the baseline accuracy and the accuracy after fine-tuning (legend located in the upper left corner). Additionally, the box plots within each violin plot display the median accuracy, while the color dots visually represent the median accuracy. The width of the violin plot conveys the frequency distribution of accuracy values. The study yielded varying levels of average accuracy improvement across different corpora. The highest average accuracy improvement of 2.48% was observed with the IndicNLP corpus, signifying substantial enhancements in model performance. Conversely, the Prothom-Alo corpus demonstrated a more modest but noteworthy improvement, with a minimum average accuracy increase of 0.75%. For the BTCC11 and OSBC corpora, the average accuracy improvements were 1.94% and 1.95%, respectively, indicating a consistent and significant boost in classification accuracy. While displaying a slightly lower average accuracy improvement of 1.26%, the BARD corpus still showcased notable progress in model performance. Collectively, the hyperparameter tuning efforts for the Bengali language

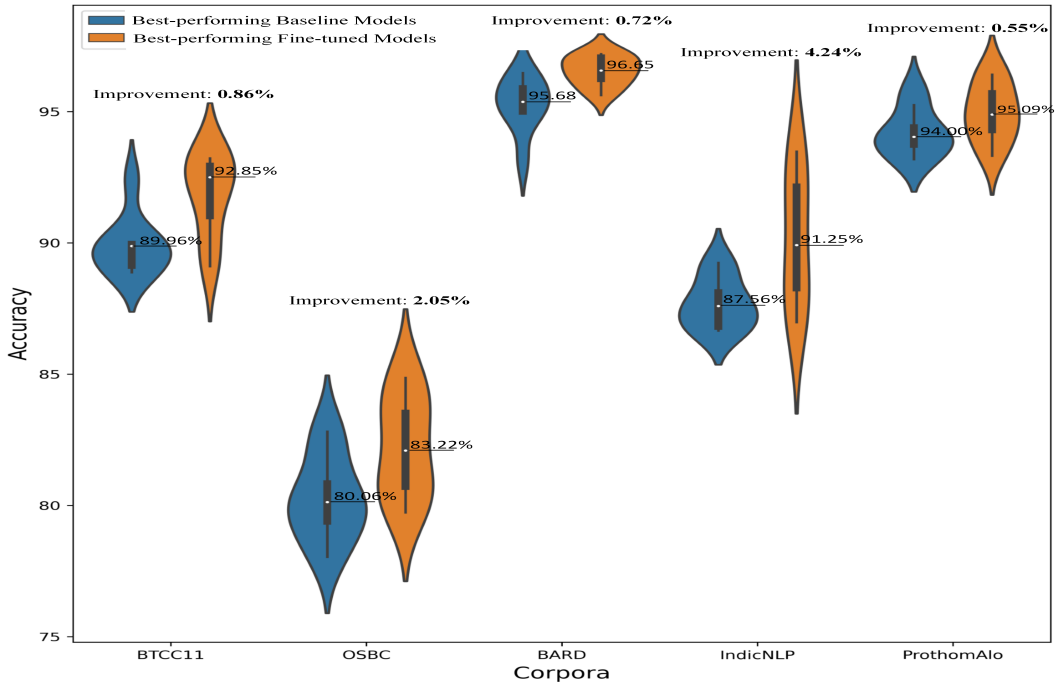


Figure 6.6: Accuracy improvement between the best-performing eight baseline and fine-tuned models across five corpora

models resulted in an overall average accuracy improvement. This improvement is attributed to refined semantic and syntactic features, enhancing the models' ability to capture intricate linguistic patterns.

Performance of Late Fusion and Best-performed Models: As outlined in Section 5.4.3.2, we utilized best-performed eight models to generate a total of 56 late fusion-based performance matrices through $\binom{8}{3}$ combinations for each corpus. Considering the involvement of five different corpora and two distinct late fusion techniques, the cumulative number of generated late fusion performance matrices reached 560. The study identified the best-performing four combinations from this extensive set of models, which are thoughtfully presented in Table 6.34.

Table 6.34: Performance of four best-performed late fusion-based techniques

Fusion-based Combination	A(%)				
	BTCC11	OSBC	BARD	Indic-NLp	Prothom-Alo
averaging(BanglaBERT, MuRIL, XLMRoBerta)	**93.31±.2**	**84.52±.2**	**97.23±.2**	**93.51±.2**	**96.43±.2**
averaging(BanglaBERT, MuRIL, DeBERTa)	93.28±.2	84.48±.2	97.21±.2	93.50±.2	96.41±.2
averaging(MuRIL, DeBERTa, XLMRoBerta)	93.18±.2	84.45±.2	97.08±.2	93.30±.2	96.21±.2
averaging(DeBERTa, XLMRoBerta, MBERT)	93.10±.2	84.06±.2	97.03±.2	93.27±.2	96.14±.2

The results deduced that the *averaging* late fusion of BanglaBERT, MuRIL,

and XLMRoBerta achieved the highest accuracy scores of 93.31±.2% for BTCC11, 84.52±.2% for OSBC, 97.23±.2% for BARD, 93.51±.2% for IndicNLP, and 96.43±.2% for Prothom-Alo, respectively, across the five corpora. This superior accuracy is attributed to the language-specific advantages of BanglaBERT and MuRIL, which yield better softmax scores when their predictions are fused, surpassing the performance of individual models. Consequently, these three models, along with the additional fourth model, DeBERTa (as indicated in Table 6.33), namely $RM_{r(1)|r(1)=BanglaBERT}$, $RM_{r(2)|r(2)=MuRIL}$, $RM_{r(3)|r(3)=XLMRoBerta}$, and $RM_{r(4)|r(4)=DeBERTa}$, have been identified as the best-performing models. They will be employed in the early fusion module.

Performance of Attention-based Early Fusion and Best-performed Models: As detailed in Section 5.4.3.3, this work generated 220 attention-based early fusion models. These models are a combination of the four best-performing models, two fusion methods, and five different corpora, resulting in a total of $5 \times 4 \times (\binom{4}{4}+\binom{4}{3}+\binom{4}{2})$ model variants. Among this extensive set of 220 models, the performance of the four best-performed models is presented in Table 6.35.

Table 6.35: Performance of four best-performing attention-based early fusion models

Model	A(%)				
	BTCC11	OSBC	BARD	IndicNLP	Prothom-Alo
$\boldsymbol{acat(MR_{r(1)}, MR_{r(2)})}$	**93.34±.2**	**85.03±.2**	**97.28±.2**	**93.60±.2**	**96.46±.2**
$acat(MR_{r(1)}, MR_{r(3)})$	93.28±.2	84.96±.2	97.24±.2	93.56±.2	96.44±.2
$acat(MR_{r(1)}, MR_{r(4)})$	93.25±.2	84.88±.2	97.19±.2	93.47±.2	96.30±.2
$acat(MR_{r(2)}, MR_{r(3)})$	93.20±.2	84.79±.2	97.16±.2	93.39±.2	96.28±.2

In Table 6.35, we can observe that the highest accuracy scores are achieved as follows: 93.34±.2% for BTCC11, 85.03±.2% for OSBC, 97.28±.2% for BARD, 93.60±.2% for IndicNLP, and 96.46±.2% for Prothom-Alo. These exceptional performance levels are attained through the attention-based early fusion method, specifically the concatenation of $MR_{r(1)}$ (BanglaBERT) and $MR_{r(2)}$ (MuRIL) models. The remarkable results can be attributed to the synergy between monolingual transformer-based language features and multilingual features, combined with attention-based feature fusion, embodied in the $cat(MR_{r(1)}, MR_{r(2)})$ fusion technique, which yields the maximum accuracy values.

To clarify further, we have utilized the statistical significance test, specifically Welch's t-test, with a significance level set at 5% (0.05). This test assesses whether the performance enhancement observed in our attention-based early fusion model holds statistical significance [167]. A summary of the statistical test results is provided in Table 6.36.

Table 6.36: Summary of statistical significance test (t-value and p-value) of four models: $cat(MR_{r(1)}, MR_{r(3)})$, $cat(MR_{r(1)}, MR_{r(4)})$, and $cat(MR_{r(2)}, MR_{r(3)})$ models with five corpora compared to $cat(MR_{r(1)}, MR_{r(2)})$

Model	p-value (Wp)/t-value (Wt)				
	BTCC11	OSBC	BARD	IndicNLP	Prothom-Alo
$acat(MR_{r(1)}, MR_{r(3)})$	0.0003/6.0	0.0006/5.6	0.003/4.0	0.004/4.0	2.7E-08/21.0
$acat(MR_{r(1)}, MR_{r(4)})$	1.8E-05/9.0	2.7E-06/12.5	1.8E-05/9.0	1.1E-06/13.0	2.3E-07/16.0
$acat(MR_{r(2)}, MR_{r(3)})$	6.5E-07/6.5	1.1E-07/19.3	2.1E-06/12.0	2.7E-08/21.0	9.3E-08/18.0

Results presented in Table 6.36 concluded that the attention-based early fusion model $cat(MR_{r(1)}, MR_{r(2)})$ exhibits statistically significant differences when compared to other attention-based early fusion models. This study conducted the Friedman test on four early fusion models (i.e., Table 6.35) across five corpora. The results of the Friedman test obtained a *Fp* of *0.0018* and *Ft* of *15.0*, which *Fp* is less than the significance level of 0.05. *This indicates that there are indeed significant differences in the performance of the attention-based early fusion model, specifically the combination of models* $MR_{r(1)}$ *and* $MR_{r(2)}$*, across the five corpora.* In this context, the term **cat**$(MR_{r(1)}, MR_{r(2)})$ is referred to as $\Delta AFuNet$.

Performance of Ablation Study and Impact Analysis:

Ablation Analysis To delve into the essential components contributing to the performance of $\Delta AFuNet$, this study conducted a component-level ablation analysis. As outlined in Section 5.4.3.4, the ablation studies focused on $\Delta AFuNet$, where we systematically varied key components, including the dropout value (Dr), the number of attention layers (NAL), and the attention layer embedding dimension (AED). Table 6.37 presents the summary of outcomes of the ablation studies.

As depicted in Table 6.37, the utmost accuracy scores were obtained, reaching 93.60% for BTCC11, 85.37% for OSBC, 97.35% for BARD, 93.74% for IndicNLP, and 96.51% for Prothom-Alo. These remarkable performance levels were explicitly attributed to fundamental component values: NAL:2, Dr:0.47, and AED:1536/1538. The ablation analysis effectively scrutinized the component-level impact of $\Delta AFuNet$, resulting in the highest accuracy for low-resource Bengali text classification corpora. The subsequent process transitioned these optimal fundamental component values to the $AFuNet$. These findings have significant implications for advancing low-resource text classification and related

Table 6.37: Summary of ablation analysis on $\Delta AFuNet$: Bold values highlight the maximum accuracy and key component configurations

CC	Valus	A(%)				
		BTCC11	OSBC	BARD	IndicNLP	Prothom-Alo
NAL	1	93.34±.2	85.03±.2	97.28±.2	93.60±.2	96.46±.2
	2	**93.60±.2**	**85.37±.2**	**97.35±.2**	**93.74±.2**	**96.51±.2**
	3	96.42±.2	85.19±.2	97.29±.2	93.68±.2	96.48±.2
Dr	0.30	93.34±.2	85.03±.2	97.28±.2	93.60±.2	96.46±.2
	0.47	**93.60±.2**	**85.37±.2**	**97.35±.2**	**93.74±.2**	**96.51±.2**
	0.50	96.42±.2	85.19±.2	97.29±.2	93.68±.2	96.48±.2
AED	1536	93.34±.2	85.03±.2	97.28±.2	93.60±.2	96.46±.2
	1536 /1536	**93.60±.2**	**85.37±.2**	**97.35±.2**	**93.74±.2**	**96.51±.2**
	1536 /1536 /768	96.42±.2	85.19±.2	97.29±.2	93.68±.2	96.48±.2

research areas.

Impact of OOV and Tokenization on AFuNet When the feature extraction process cannot locate a word within the embedding model, it is labeled as an Out-of-vocabulary (OOV) word. Conversely, if a word is deconstructed into sub-words and characters to generate a word feature through embedding, this process is called tokenization. Table 6.38 shows the effects of OOV words and the tokenization operations on AFuNet.

This research utilized three non-contextual embeddings, specifically Word2Vec, FastText, and GloVe, alongside eight transformer-based contextual embeddings. GloVe and Word2Vec cannot extract features for Out-of-vocabulary (OOV) words. In contrast, FastText utilizes character n-gram sub-word tokenization to address OOV word challenges. The definitions of OOV words, tokenized words, and matching words, as outlined in Table 6.38, are denoted as $OOVW$, TW, and MW, respectively.

Table 6.38 revealed that during the feature extraction, the highest count of Out-of-Vocabulary Words (OOVW) reached 2,434,804 in the training set and 397,820 in the test set when using GloVe embeddings with the BARD corpus. In contrast, the AFuNet model achieved impressive accuracy, with rates of 93.60% for BTCC11, 85.37% for OSBC, 97.35% for BARD, 93.74% for IndicNLP, and 96.51% for Prothom-Alo. This exceptional performance is attributed to the absence of OOV words in both the training and test phases and the tokenized sub-words effectively capturing context-aware word features. Conversely, FastText

Table 6.38: Impact of OOV words and tokenization operations on AFuNet accuracy across five corpora

Model	Corpus	OOVW (Tr/Ts)	TW (Tr /Ts)	MW (Tr/Ts)	A(%)
GloVe+CNN	BTCC11	13125 /7467	0/0	33110300 /14190166	87.43±.2
	OSBC	861357 /297820	0/0	12363977 /5395923	77.34±.2
	BARD	2434804 /397820	0/0	40615432 /18078225	94.19±0.2
	IndicNLP	89647 /19409	0/0	1750740 /440084	86.41±.2
	Prothom-Alo	1063937 /159608	0/0	29343160 /12824556	92.16±.2
FastText+CNN	BTCC11	7484 /4563	5641 /2904	33110300 /14190166	86.00±.2
	OSBC	566686 /211479	294671 /86341	12363977 /5395923	75.90±.2
	BARD	1497676 /299449	937128 /98371	40615432 /18078225	93.01±.2
	IndicNLP	64661 /14378	24986 /5031	1750740 /440084	86.19±.2
	Prothom-Alo	587629 /99896	476308 /59712	29343160 /12824556	91.96±.2
AFuNet	BTCC11	0/0	33122216 /14197317	1209 /316	**93.60±.2**
	OSBC	0/0	13223925 /5693573	1409 /170	**85.37±.2**
	BARD	0/0	43049027 /18475729	2901 /1109	**97.35±.2**
	IndicNLP	0/0	1839867 /459363	520 /130	**93.74±.2**
	Prothom-Alo	0/0	30404907 /12982923	2190 /1241	**96.51±.2**

can extract OOV features but performs poorly due to a lack of comprehensive global word statistics and the failure to consider window-based local context. AFuNet successfully addresses OOV word issues during the feature extraction process, leading to a notable enhancement in overall accuracy

Performance of AFuNet Table 6.39 provides a detailed statistical summary of AFuNet's performance across five different corpora. Results indicate that the AFuNet achieved the highest accuracy (97.35%) with the BARD corpus. This exceptional performance can be attributed to the corpus's substantial number of text samples (376,205), a relatively low number of categories (5), and fewer class imbalance issues than the other four corpora. On the other hand, the lowest accuracy, at 85.37%, was observed with the OSBC corpus. This lower performance can be attributed to the smaller number of text samples available (95,855), more significant categories (12), and more pronounced class imbalance issues than the other four corpora.

The analysis demonstrated that the performance of text classification tasks in low-resource texts (i.e., Bengali) is influenced by various factors, including the

Table 6.39: Performance summary of AFuNet

Corpus	A(%)	F1-score(%)	MA (%)			WA (%)		
			P	R	$F1$	P	R	$F1$
BTCC11	93.60±.2	92.37±.2	93	92	92	94	94	94
OSBC	85.37±.2	84.04±.2	84	84	84	85	85	85
BARD	**97.35±.2**	95.20±.2	95	95	95	97	97	97
IndicNLP	93.74±.2	91.02±.2	91	91	91	94	94	94
Prothom-Alo	96.51±.2	**95.26±.2**	95	95	95	97	97	97

number of text samples, categories, and class imbalance issues.

Comparisons among AFuNet, Baseline, Hyperparameter-tuned, and Focal Loss Models Figure 6.7 shows the Summary of comparisons that demonstrated the superior performance of the AFuNet over the baselines, hyperparameter-tuned, and focal loss models.

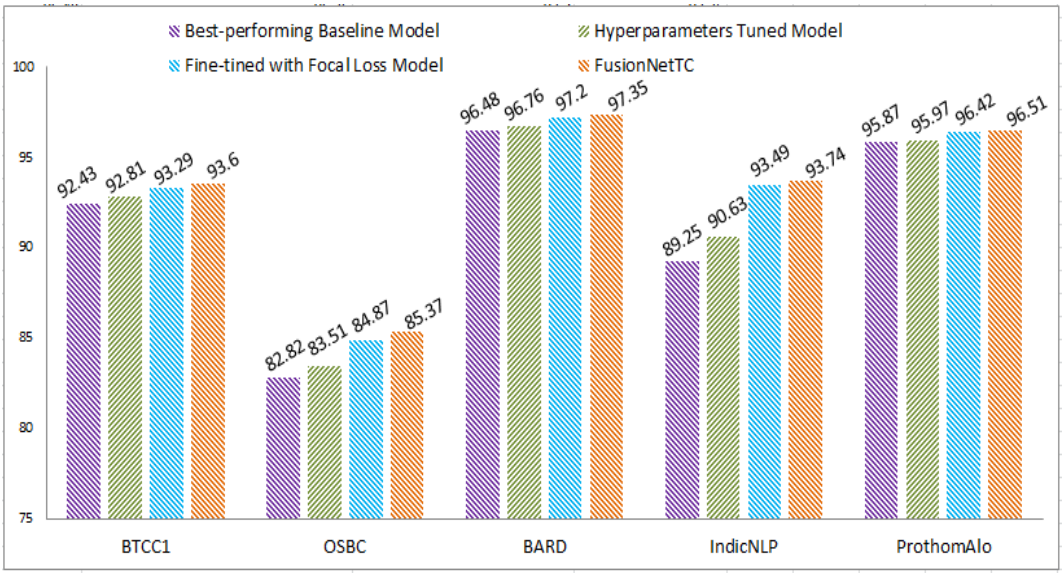


Figure 6.7: Summary of performance comparisons

Figure 6.7 reveals that the most substantial accuracy improvement of 4.49% obtained (from 89.258% to 93.74%) for the IndicNLP corpus due to three critical operations: hyperparameter tuning, focal loss implementation, and attention-based early fusion. Similarly, the BTCC11 had an improvement of 1.17%, OSBC improved by 2.55%, BARD by 0.88%, and Prothom-Alo by 0.54%. Notably, the improvements for Prothom-Alo and BARD were comparatively minor due to class balance and the considerable number of text samples in each category. The comparative analysis demonstrated that AFuNet surpassed all baseline models' performance.

Comparison between the AFuNet and Past Models In recent times, several research studies have been conducted on text classification in Bengali using machine and deep learning techniques in [164, 168, 28, 169].

Table 6.40 summarizes the comparative analysis of AFuNet and state-of-the-art text classification models in Bengali. For the sake of consistency and uniformity, all prior models and the proposed model are executed on the same machine and datasets. All five corpora are accessible; however, none previous Bengali text classification studies have shared a standardized train and test set split. Consequently, it is not feasible to directly compare accuracy with previous research. In contrast, AFuNet employs an initial random partition of the corpus into training and testing sets with a ratio of 70:30. It then utilizes the state-of-the-art method from previous Bengali text classification studies for training and inference, enabling a direct comparison of accuracy between the past state-of-the-art methods and AFuNet.

The results showed that the AFuNet achieved the highest accuracy for each corpus: 93.60% for BTCC11, 85.37% for OSBC, 97.35% for BARD, 93.74% for IndicNLP, and 96.51% for the Prothom-Alo corpus. Conversely, the lowest accuracy for the BTCC11 corpus is 88.34%, as obtained by the ELECTRA [164] model, and the OSBC corpus records a minimum accuracy of 79.45% for the AVG-M+CNN model [28]. For IndicNLP, the AC-GAN+CNN-based models achieved a minimum accuracy of 86.17%, while the Prothom-Alo corpus exhibited a minimum accuracy of 93.96%.

AVG-M+CNN and AC-GAN+CNN models [28, 169] failed to achieve superior accuracy compared to AFuNet primarily due to issues related to out-of-vocabulary (OOV) words, the absence of context-aware features, and difficulties in handling class imbalances. Conversely, ELECTRA, mBERT, and XLM-RoBerta models [164, 170, 168] fell short of obtaining better accuracy than AFuNet because of inadequately tuned hyperparameters, challenges associated with class imbalances, and a lack of fusion between monolingual and multilingual models. In summary, AFuNet outperformed all previous state-of-the-art Bengali text classification models due to the incorporation of *tuned hyperparameters, addressing OOV issues, implementing focal loss, and successfully fusing monolingual and multilingual models.*

6.5.5.3 Performance of Attention-based Early Fusion for Bengali Authorship Attribution

In this section, we present the performance evaluation of the baseline method and AuthorNet using four different corpora. The baseline method's performance

Table 6.40: Summary of comparison between the AFuNet and previous text classification models in Bengali

Model	Corpus	A (%)
Proposed (AFuNet)	BTCC11	**93.60±.2**
	OSBC	**85.37±.2**
	BARD	**97.35±.2**
	IndicNLP	**93.74±.2**
	Prothom-Alo	**96.51±.2**
ELECTRA [164]	BTCC11	88.34±.2
	OSBC	78.50±.2
	BARD	93.05±.2
	IndicNLP	87.63±.2
	Prothom-Alo	**94.18±.2**
mBERT[170]	BTCC11	88.90±.2
	OSBC	79.58±.2
	BARD	93.15±.2
	IndicNLP	86.78±.2
	Prothom-Alo	**93.89±.2**
XLMRoBerta [168]	BTCC11	92.81±.2
	OSBC	83.51±.2
	BARD	**96.76±.2**
	IndicNLP	90.63±.2
	Prothom-Alo	95.97±.2
AVG-M+CNN [28]	BTCC11	90.07±.2
	OSBC	79.45±.2
	BARD	**95.92±.2**
	IndicNLP	87.44±.2
	Prothom-Alo	93.10±.2
AC-GAN+CNN[169]	BTCC11	89.68±.2
	OSBC	80.09±.2
	BARD	**96.61±.2**
	IndicNLP	86.17±.2
	Prothom-Alo	92.96±.2

is discussed, followed by an in-depth analysis of AuthorNet's performance in the subsequent subsection.

Baseline Methods Performance Table 6.41 shows the baseline methods performances, i.e., nine deep learning-based and eight transformer-based models. Three non-contextual embeddings (i.e., GloVe, FastText & Word2Vec) and three deep learning classification methods (i.e., CNN, LSTM & GRU) combinedly produced nine classification models. Among these nine models, CNN+GloVe obtained a maximum accuracy of 93.42 for the BACC-18 corpus, 95.03% for the BAAD16 corpus, 98.63% for the LD corpus, and 94.30% for the BAAD6 corpus.

In comparison, the minimum accuracies are obtained by the GRU+Word2Vec model for the four corpora. The GRU+Word2Vec model cannot capture the local and global stylometric features due to the lack of syntactic features of the Word2Vec model, and the GRU classifier does not carry the global features.

Table 6.41: Best-performed baseline methods Accuracy (A) and F1-score (F1) of four corpora, Bold values indicate the best-performed models

	BACC-18(%)		BAAD16(%)		LD(%)		BAAD6(%)	
Model	A	F1	A	F1	A	F1	A	F1
CNN+GloVe	**93.42**	**93.00**	**95.03**	**95.00**	**98.63**	**99.00**	**94.30**	**93.75**
LSTM+GloVe	91.03	90.78	92.15	91.00	97.05	97.00	92.90	93.00
GRU+GloVe	90.87	91.00	91.59	92.00	97.12	97.00	91.21	91.00
CNN+FastText	92.64	92.12	94.60	94.01	98.63	99.00	93.78	93.00
LSTM+FastText	90.34	90.00	91.08	90.59	96.89	97.00	92.02	92.00
GRU+FastText	90.10	90.00	90.49	90.80	96.98	97.00	90.79	90.62
CNN+Word2Vec	92.04	92.00	93.76	93.53	98.33	99.00	93.17	92.95
LSTM+Word2Vec	90.19	89.86	90.00	90.00	96.15	96.00	91.50	91.00
GRU+Word2Vec	89.96	90.00	90.04	90.00	96.63	97.00	90.01	90.00
Bangla-BERT	**96.23**	**96.00**	**92.67**	**93.00**	**99.68**	**99.00**	**96.68**	**97.00**
B-ELECTRA	84.37	84.00	75.44	75.00	95.73	96.00	91.15	91.00
M-BERT	93.38	93.00	91.06	91.00	96.93	97.00	92.13	92.00
XML-Roberta	94.95	95.00	93.59	94.00	97.20	97.00	93.18	93.00
Distil-BERT	93.06	93.00	89.30	89.00	94.93	95.00	92.15	92.00
DeBERTa	94.68	95.00	78.03	78.00	**99.68**	**99.00**	95.15	95.00
Indic-BERT	84.30	84.00	73.30	73.00	85.86	86.00	92.33	92.00
MuRIL	95.23	95.00	78.14	78.00	99.33	99.00	96.33	96.00

From Table 6.41, eight transformer-based models are fine-tuned for four corpora. Among eight models, two are monolingual, i.e., Bengali language specific (Bangla-BERT & B-ELECTRA), and the remaining 6 are multilingual language models. All the models used default hyperparameters, i.e., maximum sequence length 300, batch size 8, and number of epochs 10. Using these default hyperparameters, obtained the maximum accuracy of 96.23% for the BACC-18 corpus, 92.67% for the BAAD16 corpus, 99.68% for the LD corpus, and 96.68% for the BAAD6 corpus. At the same time, the Indic-BERT, B-ELECTRA, and Distil-BERT obtained the minimum performances for four corpora due to a shortage of vocabulary in language models and failed to capture the global stylometric features of author texts.

AuthorNet Performance The performance evaluation of AuthorNet is conducted in four different aspects. Firstly, the Hyperparameters Tuning Performance examines the impact of key hyperparameters on the model's performance.

Secondly, the Best-performed Model Selection Performance assesses the individual performances of uni-modal models. It selects the best-performing monolingual and multi-lingual models for attention-based early fusion. Thirdly, the Attention-based Early Fusion Performances evaluate the performance of the fused models using attention-based mechanisms. Fourthly, Ablation Study, i.e., the impact of different components of AuthorNet. The following paragraph provides a detailed description of each of these evaluation aspects.

Hyperparameters Tuning Performance: Based on the findings presented in Table 6.41, the hyperparameter tuning phase of AuthorNet focused on evaluating the performance of 8 different transformer-based language models based on their accuracy (A) and F1-score (F1). Given the specific characteristics and limited vocabulary of Bengali language models, this study placed emphasis on optimizing three crucial hyperparameters: batch size, maximum sequence length, and epoch number. Initial experiments involved varying the batch size among 6, 8, 12, the epoch values among 10, 15, 20, and the maximum sequence lengths among 200, 300, 400. Each transformer-based model produced a total of 27 different models for each corpus, resulting from different combinations of these hyperparameters. Due to time limitations, the hyperparameters were tuned specifically on the BAAD16 corpus, and the optimized values were then applied to the remaining corpora (BACC-18, LD, and BAAD6). Figure 6.8 visually represents the minimum accuracy achieved across the 8 transformer-based models for each of the four corpora. Notably, the minimum accuracies were obtained with 10 epochs, a maximum sequence length of 200, and a batch size of 12.

Based on the observations in Figure 6.8, it can be concluded that the performance of the 8 transformer-based models has significantly improved by reducing the batch size from 12 to a smaller value. This adjustment effectively mitigated the issue of gradient overshooting caused by larger batch sizes (i.e., 12) when training on smaller samples. Furthermore, it was observed that the BAAD16 corpus exhibited relatively poor performance across the 8 transformer-based models when using a maximum sequence length of 200. This is likely due to the limited availability of stylometric features within the top 200 words of the corpus.

Figure 6.9 illustrates the accuracy of the best-performing models among the 8 transformer-based models that were fine-tuned. These models achieved their maximum accuracy when trained for 15 epochs, with a maximum sequence length of 400 and a batch size of 4. In cases where an author's text exceeded 400 words, only the top 400 words were considered, while for shorter texts, padding was applied with zeros.

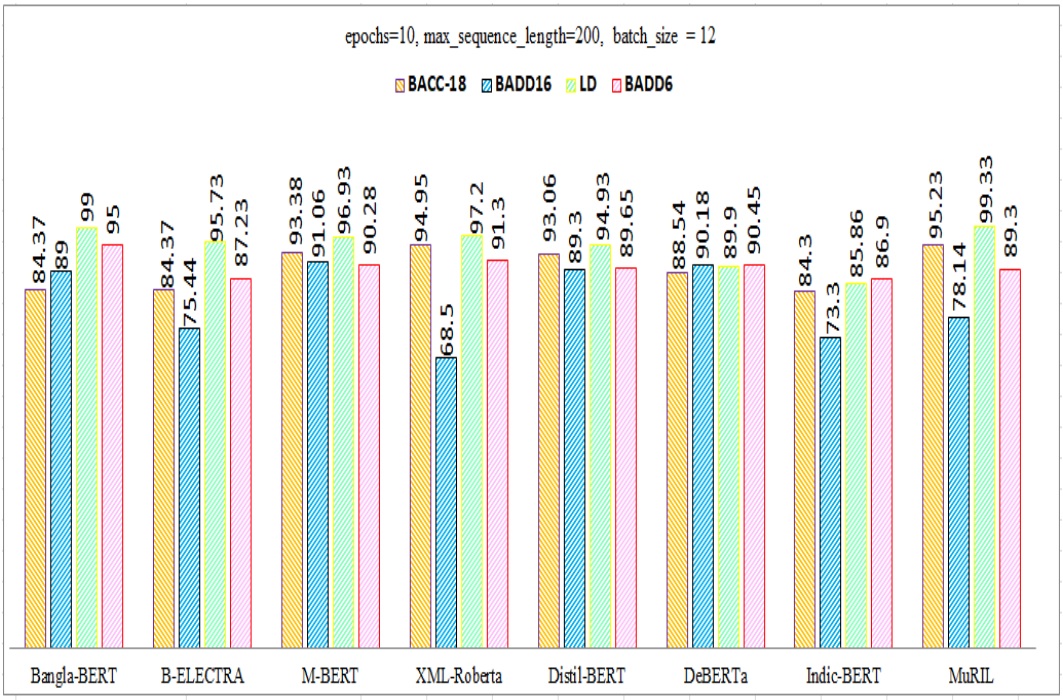


Figure 6.8: Minimum performed hyperparameters performance of 8 transformer models with 4 corpora

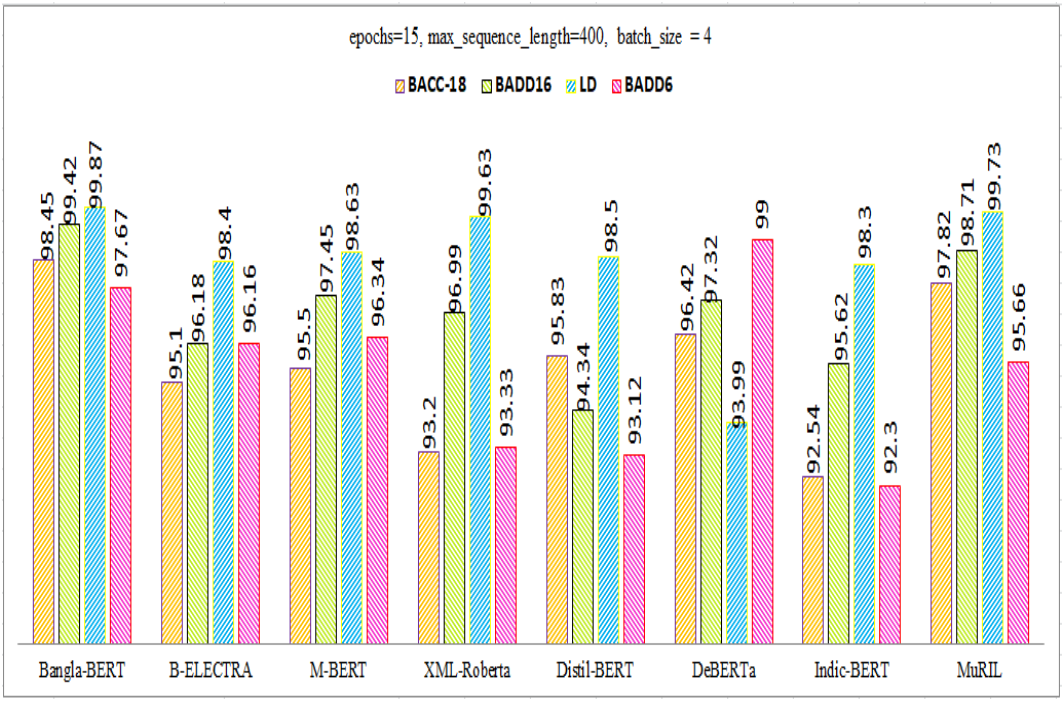


Figure 6.9: Maximum performed hyperparameters performance of 8 transformer models with 4 corpora

Figure 6.9 depicts the maximum accuracies achieved by the fine-tuned models. The monolingual Bangla-BERT-based models demonstrated the highest accuracies, with 98.45% for BACC-18, 99.42% for BAAD16, 99.87% for LD, and

97.67% for BAAD6. On the other hand, the multilingual Indic-BERT-based models achieved minimum accuracies of 92.54% for BACC-18, 95.62% for BAAD16, 98.3% for LD, and 92.3% for BAAD6. It is worth noting that the Bengali-language-based ELECTRA (B-ELECTRA), multilingual Distil-BERT, and Indic-BERT-based models exhibited lower accuracies due to the smaller number of trainable parameters compared to the distributions of the training samples. In conclusion, the monolingual Bangla-BERT-based models outperformed the multilingual models in terms of accuracy.

Best-performed Model Selection Performance: In order to determine the best-performing monolingual and multilingual transformer-based models, this study evaluated the performance of 8 models across four corpora. Table 6.42 presents the performance metrics, including accuracy (A), F1-score (F1), macro-average recall (mAr), and weighted average recall (wAr), for the four corpora and the eight fine-tuned transformer-based models.

In Table 6.42, the best-performed models are highlighted in bold. The monolingual Bangla-BERT fine-tuned model achieved the highest performance across all four corpora, making it the top-performing model. The second best-performing models were the multilingual fine-tuned MuRIL models, which performed well on three corpora: BACC-18, BAAD16, and LD. For the BAAD6 corpus, the second-best performance was achieved by the multilingual fine-tuned M-BERT model. The Bengali domain-specific transformers B-ELECTRA did not achieve better accuracy due to limitations in vocabulary and trainable parameters [28]. Similarly, DeBERTa, XLM-Roberta, and M-BERT did not achieve better accuracy due to a lack of stylometric features and vocabulary. Overall, the two top-performing models, i.e., monolingual Bangla-BERT and multilingual MuRIL, will be further used in the attention-based early fusion stage.

Attention-based Early Fusion Performance: In this study, we explored two feature-level early fusion techniques: concatenation and averaging. Based on the best-performing models (i.e., Bangla-BERT and MuRIL), we constructed four early fusion-based authorship classification models: concatenation of Bangla-BERT and MuRIL (Cat(.,)), averaging of Bangla-BERT and MuRIL (Avg(.,)), averaging followed by the attention of Bangla-BERT and MuRIL (AAvg(.,)), and concatenation followed by attention-based early fusion of Bangla-BERT and MuRIL (i.e., proposed model). The performance of these attention-based early fusion models can be seen in Table 6.43.

From Table 6.43, the proposed AuthorNet, utilizing attention-based early fu-

Table 6.42: Statistical performance summary of 8 transformer-based fine-tuned models with 4 corpora, Bold values indicate the best and second best-performed models

Model	Corpus	A(%)	F1(%)	M_{ar}(%)	WA_r(%)
Bangla-BERT	BACC-18	**98.45±0.2**	**96.00**	**99.00**	**98.00**
	BAAD16	**99.42±0.2**	**99.00**	**99.00**	**99.00**
	BAAD6	**97.67±0.2**	**98.00**	**98.00**	**98.00**
	LD	**99.87±0.2**	**100.00**	**100.00**	**100.00**
B-ELECTRA	BACC-18	95.10±0.2	95.00	89.00	95.00
	BAAD16	96.18±0.2	96.00	95.00	96.00
	BAAD6	96.16±0.2	96.00	96.00	95.00
	LD	98.40±0.2	98.00	98.00	98.00
M-BERT	BACC-18	95.50±0.2	96.00	95.00	96.00
	BAAD16	97.45±0.2	97.00	97.00	96.00
	BAAD6	**96.34±0.2**	**96.00**	**95.00**	**96.00**
	LD	98.63±0.2	99.00	99.00	98.00
XLM-Roberta	BACC-18	93.20±0.2	93.00	86.00	93.00
	BAAD16	96.99±0.2	97.00	96.00	97.00
	BAAD6	93.33±0.2	98.00	98.00	98.00
	LD	99.63±0.2	100.00	100.00	100.00
DistilBERT	BACC-18	95.83±0.2	96.00	96.00	96.00
	BAAD16	94.34±0.2	94.00	94.00	93.00
	BAAD6	93.12±0.2	93.00	92.00	93.00
	LD	98.50±0.2	99.00	99.00	100.00
DeBERTa	BACC-18	96.42±0.2	96.00	95.00	96.00
	BAAD16	97.32±0.2	97.00	96.00	96.00
	BAAD6	93.99±0.2	94.00	94.00	93.00
	LD	99.00±0.2	100.00	100.00	100.00
Indic-BERT	BACC-18	92.54±0.2	93.00	93.00	94.00
	BAAD16	95.62±0.2	96.00	96.00	97.00
	BAAD6	92.30±0.2	92.00	92.00	93.00
	LD	98.3±0.2	98.00	98.00	99.00
MuRIL	BACC-18	**97.82±0.2**	**98.00**	**98.00**	**98.00**
	BAAD16	**98.71±0.2**	**99.00**	**99.00**	**99.00**
	BAAD6	95.66±0.2	97.00	97.00	98.00
	LD	**99.73±0.2**	**100.00**	**100.00**	**100.00**

sion, demonstrated improvements in accuracy compared to the best-performed Bangla-BERT models. Specifically, the proposed model (i.e., AuthorNet) achieved a 0.44% accuracy improvement for the BACC-18 corpus (from 98.45% to 98.86%), 0.07% for the BAAD16 corpus (from 99.42% to 99.49%), and 0.24% for the BAAD6 corpus (from 97.67% to 97.91%). However, there was no improvement in accuracy for the LD corpus. On the other hand, the average fusion model showed a decrease in accuracy, as it could not effectively capture the complex re-

Table 6.43: Performance summary of attentional and non-attentional early fusion

Model	Corpus	A(%)	F1(%)
Cat(Bangla-BERT,MuRIL)	BACC-18	98.52±0.2	97.00
	BAAD16	99.47±0.2	99.00
	BAAD6	97.71±0.2	98.00
	LD	99.87±0.2	100.00
Avg(Bangla-BERT,MuRIL)	BACC-18	98.36±0.2	96.00
	BAAD16	99.21±0.2	99.00
	BAAD6	97.48±0.2	97.00
	LD	99.63±0.2	100.00
AAvg(Bangla-BERT,MuRIL)	BACC-18	98.43±0.2	96.00
	BAAD16	99.45±0.2	99.00
	BAAD6	97.69±0.2	98.00
	LD	99.87±0.2	100.00
AuthorNet (Proposed)	**BACC-18**	**98.86±0.2**	**98.00**
	BAAD16	**99.49±0.2**	**99.00**
	BAAD6	**97.91±0.2**	**98.00**
	LD	99.87±0.2	100.00

lationships between the individual models and leverage their respective strengths. In contrast, concatenation-based fusion successfully incorporated the distinctive features of each model and produced more generalized features. Notably, the attention operation played a crucial role in identifying the important features of each individual model, contributing to the overall performance enhancement of the attention-based early fusion approach.

Statistical Significance Test: The statistical hypothesis test known as Welch's t-test was conducted with a significance level of 5% (0.05) to determine whether the performance improvement observed in our proposed model is statistically significant when compared to baseline models [167, 171]. In this study, AuthorNet and eight transformer-based fine-tuned models were executed consecutively six times, and their respective accuracies were recorded. The accuracy values of the baseline models formed one group, while the accuracy of AuthorNet constituted another group. Welch's t-test was then performed to compare these groups, resulting in the calculation of t-values and p-values for four corpora and eight transformer-based models compared to AuthorNet. The summary of the overall t-values and p-value can be found in Table 6.44.

Based on the information presented in Table 6.44, we can draw the conclusion that the results of the proposed model are statistically significant across four corpora. However, it is worth noting that in the case of the *LD corpus*

Table 6.44: Statistical significance test t-value and p-value summary of transformer-based models with four corpora compared to AuthorNet, Bolded values indicate that the difference is statistically significant

Model	t-value/p-value			
	BACC-18	BAAD16	BAAD6	LD
Bangla-BERT	**3.65/0.02**	**11.38/6.3E-06**	**24.00/9.6E-09**	0.00/1.0
B-ELECTRA	**40.11/1.9E-06**	**523.67/4.7E-18**	**176.00/1.2E-15**	**202.46/3.9E-16**
M-BERT	**35.85/3.3E-06**	**254.75/1.7E-13**	**157.00/3.0E-15**	**141.55/8.7E-14**
XLM-Roberta	**61.22/3.6E-07**	**473.58/4.4E-19**	**458.00/5.7E-19**	**33.28/7.2E-10**
DistilBERT	**32.50/5.0E-06**	**742.76/5.5E-18**	**522.85/7.6E-19**	**156.46/4.2E-14**
DeBERTa	**25.88/1.2E-05**	**270.10/1.2E-13**	**23.31/1.9E-05**	**134.71/1.0E-13**
Indic-BERT	**68.65/2.4E-07**	**612.53/1.4E-18**	**611.23/2.2E-19**	**179.40/1.5E-14**
MuRIL	**10.52/0.00045**	**97.25/6.1E-11**	**225.00/1.7E-16**	**15.37/8.2E-07**

with Bangla-BERT models, the statistical significance compared to the proposed AuthorNet is not observed.

Ablation Studies: To address **RQ4**, this study conducted ablation studies in two different scenarios: (i) model-level ablation and (ii) AuthorNet component-level ablation. The model-level ablation analysis aimed to identify the top-performing models among the models evaluated with four corpora. Conversely, the AuthorNet component-level ablation studies investigated the influence of specific components, such as maximum sequence length, dropout values, number of attention layers, and attention layer embedding size, on the accuracy of the AuthorNet model.

Model-level Ablation: Model-level ablation analysis in authorship attribution involves systematically evaluating the performance of different models by selectively disabling or removing specific models and assessing the impact on attribution accuracy. The performance of three deep learning models (CNN, LSTM, and GRU) and eight transformer-based language models (i.e., Bangla-BERT, B-ELECTRA, M-BERT, XML-Roberta, DistilBERT, DeBERTa, Indic-BERT and MuRIL) on four corpora using default hyperparameters is shown in Table 6.41. The results of the ablation analysis reveal that the transformer-based language models outperformed the deep learning models across all four corpora. In the subsequent phase, ablation analysis was conducted on the eight transformer-based models using tuned hyperparameters for the four corpora, and the findings are summarized in Table 6.42. Based on the model-level ablation analysis presented in Table Table 6.41 and 6.42, the two best-performing transformer-based models for the four corpora are Bangla-BERT and MuRIL. These models are then used to create attention-based early fusion models, collectively called AuthorNet.

AuthorNet Component-level Ablation: Table 6.45 provided the results of component-level ablation studies for AuthorNet. These studies analyze the effects of various component configurations (CC), including MSL (Maximum Sequence Length), DV (Dropout Values), NAL (Number of Attention Layers), and ALES (Attention Layer Embedding Size), across four corpora: BACC-18, BAAD16, BAAD6, and LD.

Table 6.45: AuthorNet component-level ablation analysis summary

CC	Valus	A(%)			
		BACC-18	BAAD16	BAAD6	LD
MSL	100	96.65±.1	97.23±.1	95.48±.1	99.23±.1
	200	97.02±.1	97.83±.1	95.67±.1	99.63±.1
	300	98.02±.1	98.12±.1	96.07±.1	99.63±.1
	400	**98.86±.1**	**99.49±.1**	**97.91±.1**	**99.87±.1**
DV	0.50	96.65±.1	97.23±.1	95.48±.1	99.23±.1
	0.45	**98.86±.1**	**99.49±.1**	**97.91±.1**	**99.87±.1**
	0.55	97.02±.1	97.83±.1	95.67±.1	99.63±.1
	0.60	98.02±.1	98.12±.1	96.07±.1	99.63±.1
NAL	1	96.65±.1	97.23±.1	95.48±.1	99.23±.1
	2	**98.86±.1**	**99.49±.1**	**97.91±.1**	**99.87±.1**
	3	98.02±.1	98.12±.1	96.07±.1	99.63±.1
	3	97.02±.1	97.83±.1	95.67±.1	99.63±.1
ALES	1536	96.65±.1	97.23±.1	95.48±.1	99.23±.1
	1536/1536	**98.86±.1**	**99.49±.1**	**97.91±.1**	**99.87±.1**
	1536/1536/1536	98.02±.1	98.12±.1	96.07±.1	99.63±.1
	1536/1536/768	97.02±.1	97.83±.1	95.67±.1	99.63±.1

From Table 6.45, the highest accuracies recorded were 98.86% for BACC-18, 99.49% for BAAD16, 97.91% for BAAD6, and 99.87% for LD. These superior results were achieved using an optimal component configuration consisting of an MSL value of 400, a DV value of 0.45, an NAL value of 2, and an ALES configuration of 1536/1536. This configuration, featuring a longer maximum sequence length, lower dropout rate, two attention layers, and a specific attention layer embedding size, likely contributed to improved performance by facilitating better generalization, attentiveness, and information representation within the model.

Overall, the model-level ablation analysis provided valuable insights into the effectiveness of different models across the four corpora. Additionally, the component-level analyses of AuthorNet offered comprehensive insights into the performance and configurations of individual components. These findings contribute to a deeper understanding of the underlying factors influencing the accuracy of AuthorNet, ultimately informing the development of more effective and efficient sys-

tems for resource-constrained authorship attribution. The results of this study, thus, hold significant implications for advancing the field of low-resource authorship attribution and related research areas.

Impact of Out-of-vocabulary and Tokenization on AuthorNet's Accuracy: Table 6.46 provides insights into the impact of out-of-vocabulary (OOV) words and tokenization operations on the accuracy of AuthorNet in authorship attribution. The table includes the number of OOV words (#OOV), the number of tokenized words (#TkW), the number of matched words (#MtW), and the accuracy (A(%)). OOV words represent the words that are not present in the embeddings, while tokenization is applied when an embedding or language model is unable to find a word in the pre-trained vocabulary, breaking the word into sub-words or characters to find a match with embeddings.

The results show that the non-contextual GloVe model has a significant number of OOV words, with 85,342 for BACC-18, 107,804 for BAAD16, 47,350 for BAAD6, and 13,062 for LD. Similarly, the non-contextual and sub-word-based FastText model has 32,562, 68,310, 16,938, and 5,045 OOV words for the respective corpora. Despite employing tokenization for handling OOV words, these models struggle to achieve higher accuracy compared to AuthorNet, ranging from 92.46% to 98.63%.

On the other hand, the context-aware models, Bangla-BERT and MuRIL, show no OOV words for any of the corpora, indicating their effectiveness in handling all the words through tokenization. Bangla-BERT achieves accuracies ranging from 97.67% to 99.87%, while MuRIL achieves accuracies ranging from 95.66% to 99.73%. These models highlight the importance of robust language models that capture contextual and semantic features for improved accuracy in authorship attribution tasks.

The proposed AuthorNet, which is an attentional early fusion model, also shows no OOV words across the corpora. This implies that the model can effectively handle the vocabulary present in the datasets. The accuracy of AuthorNet ranges from 97.91% to 99.49%, outperforming the non-contextual embedding and non-fusion models in most cases. This highlights the effectiveness of attentional early fusion and its ability to capture the complex relationships between the models and leverage their strengths for improved accuracy.

Overall, the results suggest that the presence of OOV words can have varying effects on the performance of authorship attribution models. Attentional early fusion models, such as AuthorNet, demonstrate better adaptability to different vocabularies, resulting in higher accuracies. The ability to handle OOV words

Table 6.46: Impact of OOV words and tokenization operation on AuthorNet accuracy

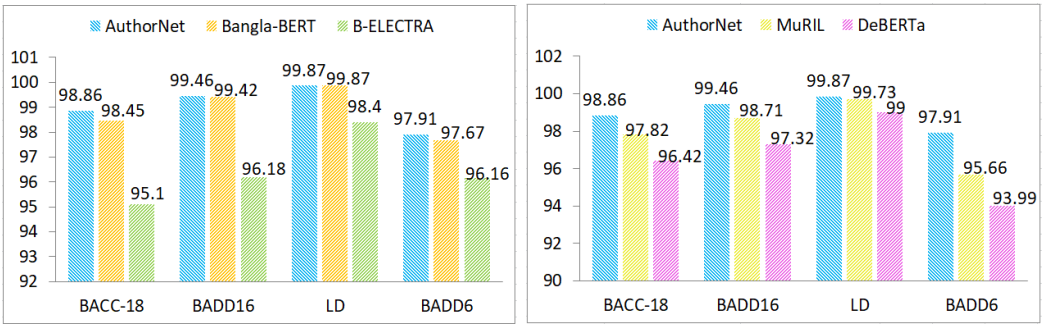
Model	Corpus	#OOV	#TkW	#MtW	A(%)
GloVe+CNN	BACC-18	85,342	0	294,658	93.42±0.2
	BAAD16	107,804	0	482,196	95.03±0.2
	BAAD6	47,350	0	122,650	94.30±0.2
	LD	13,062	0	66,938	98.63±0.2
FastText+CNN	BACC-18	32,562	52,780	294,658	92.46±0.2
	BAAD16	68,310	39,494	482,196	94.60±0.2
	BAAD6	16,938	30,412	122,650	93.78±0.2
	LD	5,045	8,017	66,938	98.63±0.2
Bangla-BERT	BACC-18	0	379,040	960	98.45±0.2
	BAAD16	0	589,003	997	99.42±0.2
	BAAD6	0	169,350	650	97.67±0.2
	LD	0	79,620	380	99.87±0.2
MuRIL	BACC-18	0	379,347	653	97.82±0.2
	BAAD16	0	589,279	721	98.71±0.2
	BAAD6	0	169,587	413	95.66±0.2
	LD	0	79,730	270	99.73±0.2
AuthorNet	**BACC-18**	0	**758,387**	**1,613**	**98.86±0.2**
	BAAD16	0	**1,178,28**	**21,718**	**99.49±0.2**
	BAAD6	0	**338,937**	**1,063**	**97.91±0.2**
	LD	0	**159,350**	**650**	**99.87±0.2**

and perform accurate tokenization is crucial in achieving superior performance in authorship attribution tasks.

6.5.6 Comparison of AuthorNet with Monolingual and Multilingual Approaches

To assess the effectiveness of the proposed system, this study conducted a comprehensive comparison between AuthorNet and several monolingual and multilingual transformer-based models specifically designed for Bengali authorship attribution. The monolingual models include Bangla-BERT and B-ELECTRA, while the multilingual models comprise M-BERT, XLM-RoBERTa, DistilBERT, DeBERTa, Indic-BERT, and MuRIL. The accuracy results of the two best-performing monolingual models, as well as AuthorNet, are presented in Figure 6.10a, while Figure 6.10b illustrates the performance of the two selected multilingual models in comparison to AuthorNet. Figure 6.10 provides an overall visualization of the accuracy comparisons between AuthorNet and the top-performing monolingual and multilingual models.

From the analysis presented in Figure 6.10b, it is evident that AuthorNet



(a) Comparison of AuthorNet and Monolingual Best-2 Models Accuracy

(b) Comparison of AuthorNet and Multilingual Best-2 Models Accuracy

Figure 6.10: Monolingual and Multilingual Best-2 Models Accuracy Comparison with the Proposed AuthorNet

achieves a remarkable accuracy improvement of 3.76% for the BACC-18 corpus compared to the monolingual models. Specifically, AuthorNet achieves an accuracy of 98.86%, while B-ELECTRA attains an accuracy of 95.10%. Similarly, Figure 6.10b highlights the superior performance of AuthorNet over the multilingual models, with a maximum accuracy improvement of 3.92% for the BAAD6 corpus. In this case, AuthorNet achieves an accuracy of 97.91% compared to DeBERTa, which achieves an accuracy of 93.99%. The notable enhancements in accuracy can be attributed to the attention-based early fusion employed by AuthorNet, enabling the capture of fine-grained semantic information at the word level and author-specific patterns at the sentence and paragraph levels. Based on these compelling results, it is evident that the proposed AuthorNet model surpasses the state-of-the-art monolingual and multilingual transformer-based language models, establishing its superiority in Bengali authorship attribution tasks.

6.5.7 Comparison of the Proposed AuthorNet with Existing Approaches

In this study, we assess the performance of the proposed AuthorNet model in comparison to existing approaches using four benchmark corpora for Bengali authorship attribution: BACC-18, BAAD16, BAAD6, and LD. Previous studies in this field have employed various traditional feature extraction techniques, including TF-IDF, NB, GloVe, FastText, and Word2Vec. Additionally, both traditional and deep learning-based attribution methods such as SVM, LSTM, ULMFiT, and CNN have been utilized [101, 128, 150, 99, 36]. Table 6.47 presents the accuracy results obtained for the four benchmark corpora using the proposed AuthorNet

model and the existing models. The existing models were trained and tested using the default train-test partition and their respective hyperparameters, as mentioned in the references [36, 172, 101, 128, 150, 99]. It is noteworthy that the AuthorNet model consistently outperforms the existing approaches across all four corpora.

Table 6.47: Performance compression of proposed AuthorNet and exiting approaches

Method	A(%)			
	BACC-18	BAAD16	BAAD6	LD
Proposed(AuthorNet)	**98.86±0.2**	**99.49±0.2**	**97.91±0.2**	**99.87±0.2**
GloVe+LSTM [172]	92.53±0.2	93.93±0.2	91.45±0.2	98.51±0.2
GloVe+CNN [101]	93.45±0.2	95.05±0.2	93.08±0.2	98.67±0.2
TF+NB [128]	65.17±0.2	62.44±0.2	70.45±0.2	98.00±0.2
GloVe+SVM [150]	68.00±0.2	64.21±0.2	83.00±0.2	94.66±0.2
Fasttext+CNN [99]	79.44±0.2	76.82±0.2	75.45±0.2	97.60±0.2
TL-ULMFiT [36]	95.97±0.2	98.40±0.2	95.33±0.2	99.63±0.2

The proposed AuthorNet model demonstrates significant improvements in performance compared to the best-performing existing approach, TL-ULMFiT [36], across various Bengali authorship attribution corpora. Specifically, there is a notable increase of 2.92% for the BACC-18 corpus, 2.58% for the BAAD6 corpus, 1.09% for the BAAD16 corpus, and 0.24% for the LD corpus. It is worth mentioning that TL-ULMFiT utilized the entire token of the BAAD16 and BAAD6 corpora, while AuthorNet employed only the first 400 words from each author sample. Despite this difference, the proposed AuthorNet consistently outperforms the previous approaches regarding accuracy for all four corpora. This improvement can be attributed to the early fusion and attentional operations implemented in AuthorNet, which effectively capture author-level stylometric, semantic, and syntactic features necessary for accurate authorship attribution. Based on these results, it can be concluded that AuthorNet surpasses all existing methods in Bengali authorship attribution.

6.5.8 Extrinsic Evaluations: Text Classifiers Performance

To provide a comprehensive performance analysis of the resource-constrained text classification system, we have compiled in Table 6.48. The research initially starts with non-contextual embeddings coupled with a deep learning-based classification system. As the study progressed, transformer-based models were introduced, culminating in the proposal of attention-based early fusion networks.

Table 6.48: Summary of top-performing text classification across different corpora

Tasks	Corpus	Model	A(%)
Authorship Attribution	BACC-18 BAAD16 LD	GloVe+CNN	95.02±0.2 93.45±0.2 98.67±0.2
Document Classification	BDTCC	GloVe+VDCNN	96.96±0.2
Document Classification	BTCC11 Prothom-Alo IndicNLP BARD	AVG-M+CNN	90.07±0.2 93.10±0.2 87.44±0.2 95.92±0.2
Bengali Covid Text Classification	BCoVC	CovTiNet (APeAGF+ACNN)	95.92±0.2
Arabic Covid Text Classification	AraCoV	AraCovTexFinder (AraBERT, BERT-Arabic, DeBERTa-V3)	98.89±0.2
Document Classification	BTCC11 OSBC BARD IndicNLP Prothom-Alo	Proposed (AFuNet)	93.60±0.2 85.37±0.2 97.35±0.2 93.74±0.2 96.51±0.2
Authorship Attribution	BACC-18 BAAD16 BAAD6 LD	AuthorNet (AFuNet)	98.86±0.2 99.49±0.2 97.91±0.2 99.87±0.2

As depicted in Table 6.48, the initial optimal performance yielded peak accuracies of 95.02% for BACC-18, 93.45% for BAAD16, and 98.67% for LD corpus using the GloVe with CNN classification model (GloVe+CNN). Following the implementation of attention-based early fusion with transformer-based language models (AuthorNet), the overall accuracies reached 98.86% for BACC-18, 99.49% for BAAD16, 97.91% for BAAD6, and 99.87% for LD. Notably, AuthorNet surpassed all baseline models, showcasing accuracy improvements ranging from 0.24% to 2.92% across the four corpora.

In the initial phase of this study, the top-performing deep learning-based model (AVG-M+CNN) achieved peak accuracies of 90.07% for BTCC11, 93.10% for Prothom-Alo, 87.44% for IndicNLP, and 95.92% for BARD. Meanwhile, in the final stage, the proposed attention-based early fusion of transformers model (AFuNeT) reached maximum accuracy of 93.60% for BTCC11, 85.37% for OSBC, 97.35% for BARD, 93.74% for IndicNLP, and 96.51% for Prothom-Alo. AFuNet showcased notable accuracy enhancements, ranging from 0.54% to 4.49%, un-

derscoring its effectiveness in Bengali text classification. The proposed AFuNeT has achieved state-of-the-art (SOTA) performances across authorship attribution corpora, text classification corpora, and Covid text identification corpora, showcasing its comprehensive effectiveness.

6.5.9 Multimodal Text Classification Performance

The performance of Bengali multimodal text classification was evaluated using various models, as summarized in Table 6.49. When considering text-only models, BanglaBERT achieved an accuracy of 84.23% and an F1-score of 82.89%, while the ViT model, focusing solely on image features, attained slightly higher accuracy at 85.45% and F1-score at 83.95%. However, the integration of both text and image features using the ViT+BanglaBERT fusion model significantly improved classification performance, yielding an accuracy of 89.68% and an F1-score of 88.30%. This demonstrates the effectiveness of leveraging multimodal

Model	Accuracy (%)	F1-score (%)
BanglaBERT(only text)	84.23	82.89
ViT (only image)	85.45	83.95
ViT+BanglaBERT (Fusion)	89.68	88.30
BLIP (multimodal:image-text)	78.22	76.51

Table 6.49: Performance of Bengali multimodal text classification on BMMTC Corpus

information for enhanced classification accuracy. Notably, the BLIP model, utilizing a multimodal approach with both image and text inputs, exhibited lower performance compared to the fusion model, achieving an accuracy of 78.22% and an F1-score of 76.51%. These results underscore the importance of synergistically combining textual and visual modalities for improved performance in Bengali multimodal text classification tasks.

6.5.10 Error Analysis

The error analysis presented in this study provides a comprehensive examination of both the limitations and strengths both of the deep learning-based models and transformer-based models. The following subsections analyzed the error of deep learning-based models and transformer-based models.

6.5.11 Error Analysis of Deep Learning-based Models

The deep learning-based error analysis has been conducted for four types of models, i.e., (i) GloVe with CNN Model, (ii) GloVe with VDCNN Model, (iii) Meta-Embedding with CNN Model, and (iv) Attention-based Meta-Embedding with ACNN Model. The following paragraphs describe these model errors.

(i) Error Analysis of GloVe with CNN Model: It is evident from Tables 6.10 and 6.11 that GloVe+CNN is the best-performing model to classify text documents for the Bengali language. A detailed error analysis uses the confusion matrix (CM) to investigate more insights into the individual author's class performance.

Figure 6.11, 6.12, and 6.13 illustrate the confusion matrix for BACC-18, BAAD16, and LD datasets, respectively. The diagonal cells of CM denote the correctly classified numbers. The dark pink cells represent the maximum incorrectly classified numbers, whereas the light pink indicates the multiple incorrectly classified numbers for a particular author class.

Concerning BACC-18, among 32 texts of author 13, 20 texts are correctly classified, and 12 texts are incorrectly classified (e.g., maximum error), whereas all texts of author 18 and author 16 are correctly classified (e.g., minimum error). In BAAD16, among 906 texts of author 07, there are only 13 texts incorrectly classified, which is the minimum misclassified class. Among 44 texts of author 03, 16 texts are classified incorrectly, indicating the maximum misclassification rate of BAAD16. Concerning LD, among 250 texts of authors 02 and 10, five texts are incorrectly classified. On the other hand, all texts (250) of author 01 are classified correctly.

The confusion matrix concluded that a minimum error rate (0.00%) is achieved for authors 18 and 16 on the BACC-18 corpus. On the other hand, on the BAAD16 corpus, a minimum of 1.00% error rate is obtained for author 07, whereas author class 01 gained the lowest (0.00%) error rate on the LD corpus.

Fig. 6.14 shows the classifier performance with actual and predicted labels on four sample texts. The 'Yes' in the remarks column indicates that the proposed approach could correctly classify, whereas the word 'No' represents the classification failure. Input 1 considered from BACC-18, and LD, which are correctly predicted by the proposed approach (GloVe+CNN). However, GloVe+LSTM, Word2Vec+LSTM, and GloVe+SVM models have failed to classify input 1. Inputs 2 and 3 are taken from LD and BADD datasets where all models have failed to predict the actual author class. The proposed method (GloVe+CNN) and LSTM+GloVe can correctly predict the input text 4, whereas Word2Vec+LSTM

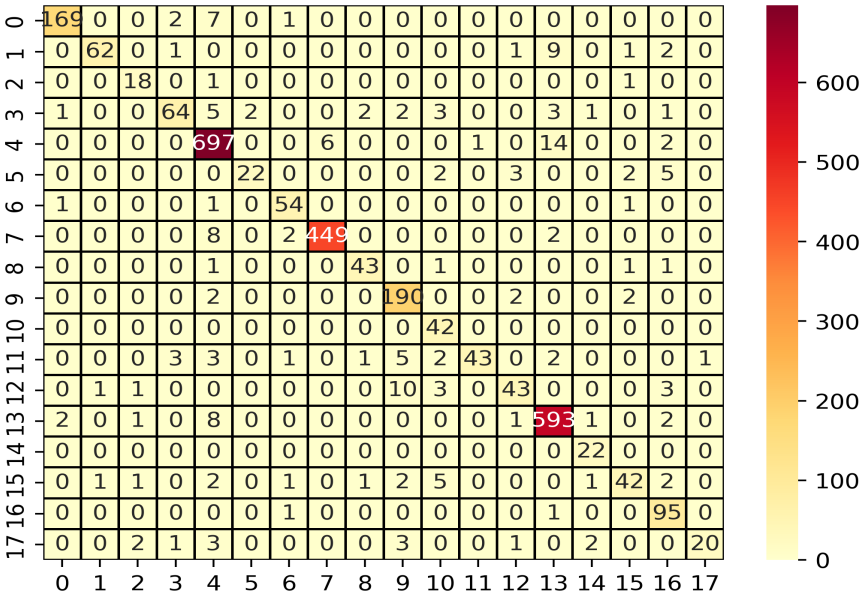


Figure 6.11: Confusion Matrix of BACC-18 and corresponding indexes are referees as *Bankim Chandra Chattopadhyay (0)*, *Emon Jubayer (1)*, *Humayun Ahmed (2)*, *Humayun Azad (3)*, *Jibanananda Das (4)*, *Kazi Nazrul Islam (5)*, *Rabindranath Tagore (6)*, *Samaresh Majumdar (7)*, *Sharadindu Bandyopadhyay (8)*, *Sarat Chandra Chattopadhyay (9)*, *Satyajit Ray (10)*, *Shamsur Rahman (11)*, *Shirshendu Mukhopadhyay (12)*, *Sukumar Ray (13)*, *Sunil Gangopadhyay (14)*, *Swami Vivekananda (15)*, *Tareque Anu (16)* and *Taslima Nasrin (17)*

and GloVe+SVM cannot predict. The error analysis concludes that the proposed approach has some limitations, such as if the text's size is too short or too long, it cannot predict the correct label. The document-level similarity between semantics and syntactic is also responsible for the misclassification.

(ii) Error Analysis of GloVe with VDCNN Model: The confusion matrix (CM) is a table that is used to summarize the performance of a classifier in tabular form [173]. Figure 6.15 shows the confusion matrix, which represents the summary of prediction results on the proposed Bengali text categorization system. The number of correct and incorrect predictions is summarized with count values and broken down by each class. This matrix visualizes the classification error/confusion when it makes predictions. The diagonal blue cell represents the number of correct predictions of texts as the actual class and other cells represent the misclassification as other classes. For example, the HE class is evaluated for a total of 859 unlabeled texts. Among these, 833 texts are correctly predicted as HE class represented in the first diagonal cell of Table 13, and 26 texts are

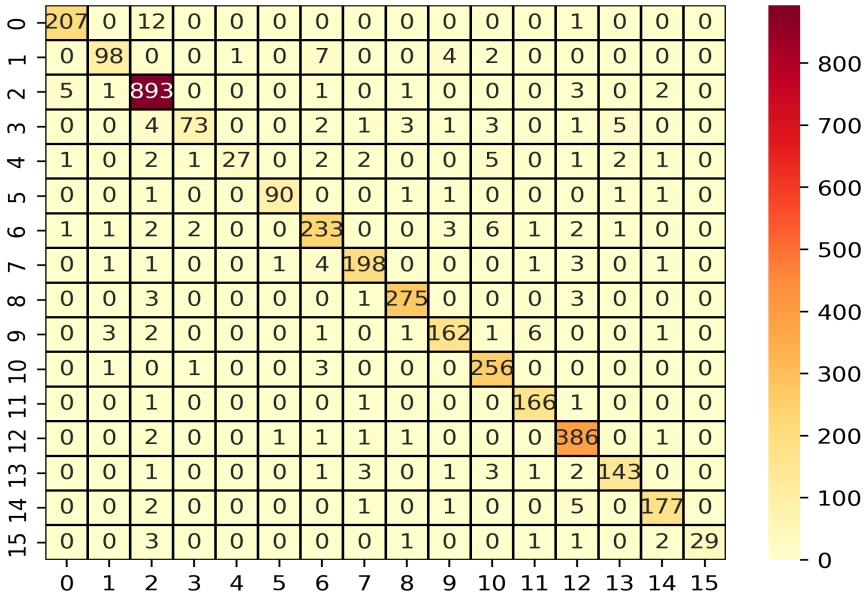


Figure 6.12: Confusion Matrix of BAAD16; corresponding indexes are referees as *Bankim Chandra Chattopadhyay (0)*, *Humayun Ahmed (1)*, *Manik Bandopadhyay (2)*, *Muhammed Zafar Iqbal (3)*, *Kazi Nazrul Islam (4*, *Nihar Ranjan Gupta (5)*, *Rabindranath Tagore (6)*, *Shirshendu Mukhopadhyay (7)*, *Samaresh Majumdar (8)*, *Shordindu Shekhor Chakma (9)*, *Sarat Chandra Chattopadhyay (10)*, *Satyajit Ray (11)*, *Sunil Gangopadhyay (12)*, *Tarasankar Bandyopadhyay (13)*, *Taslima Nasrin (14)* and *Zahir Raihan(15)*

misclassified as other classes.

The confusion matrix illustrates that the minimum misclassification rate occurred in the lifestyle category (1.00%), whereas the maximum misclassification rate was achieved in the environment category (8.99%). It complied with the confusion matrix that out of 21, 477 documents, the proposed system correctly predicted 20, 825 documents and incorrectly predicted 652 documents (Fig. 6.15). Several overlaps have occurred between categories in classifying documents. As an example, the health category correctly classified 833 documents out of 859. Out of the remaining 26 documents, nine overlapped with the LS category, and two overlapped with CR, EC, ET, PL, SP categories, three overlapped with the OP ST categories, and one overlapped with the EL category, respectively. There was no overlap with the AC, AR, and ED categories. It is apparent from the confusion matrix that more documents should be included in some categories during the training phase for improvement of performance (such as in the LS category). In between, frequent class overlap (e.g. CR and AC) can be reduced with the addition of more documents.

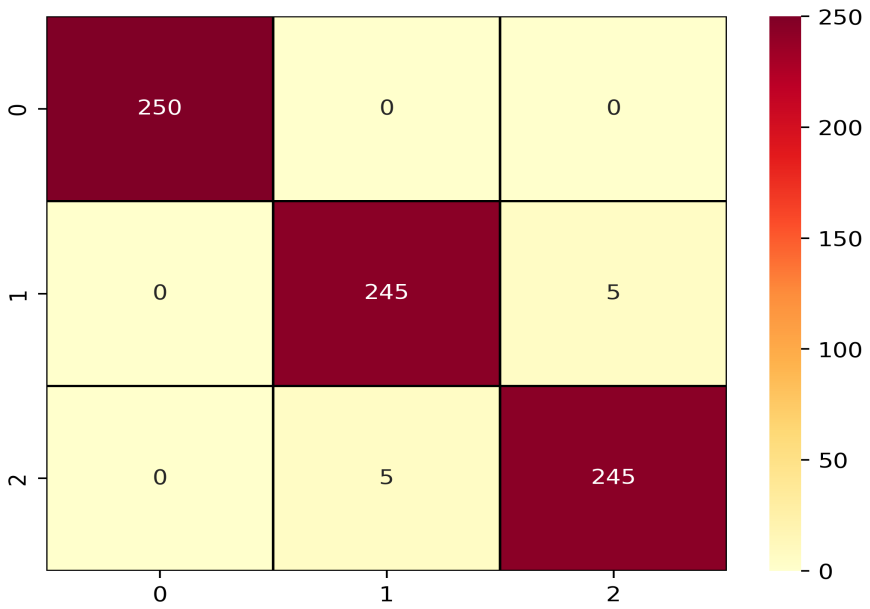


Figure 6.13: Confusion Matrix of LD and corresponding indexes are referees as *Bankim Chandra Chattopadhyay* (0), *Rabindranath Tagore* (1) and *Sarat Chandra Chattopadhyay* (2)

(iii) Error Analysis of Meta-Embedding with CNN Model: A detailed error analysis is conducted to acquire in-depth insights into the model's performance using the confusion matrix. The confusion matrix represents the actual error of the model concerning the corpus. Figs. 6.16 (a) and (b) show the BARD as well as BTCC11 corpora errors and actual scenarios. The maximum misclassified class index is 0, and the minimum misclassified class index is 3 (Fig. 6.16 (a)). The maximum misclassification occurs when the other class contains similar or common information (e.g., category index 0 & 4). Class index 3 is a more distinguished category than other categories, so the misclassification rate is minimal (e.g., 1.84%). For Fig. 6.16b, (b) the most inaccurate class index is 2, and the error rate is about 25.0% due to common feature distribution among class indexes 2, 10, and 0. Though these categories are semantically and syntactically different but have some common words, data distributions remain. In the other case, a minimum error rate of 2.0% is obtained for class index nine owning to disjoints features and data distributions.

Diagonal values in Fig. 6.17 (a) and (b) represent the correct prediction, and another cell value indicates the wrong prediction. In the confusion matrix 6.17 (a), the maximum misclassification class is 1 (e.g., 56% error) due to common category information contained between class indexes 1 and 3, respectively. The

Sample Input	Sample Output (Predicted labelled) (SoftMax Score)				Actual labelled	Remarks
	GloVe+CNN (Proposed)	GloVe+LSTM	Word2Vec+ LSTM	GloVe + SVM		
Corpus : SBD18, Type : Poem Input1 = গভীর কালো মেঘের পরে রঙিন ধনু বাঁকা রঙের তুলি বুলিয়ে মেঘে খিলান যেন আঁকা গবুজ ঘাসে রোদের পাশে আলোর কেরামতি রঙিন বেশে রঙিন ফুলে রঙিন প্রজাপতি অন্ধ মেয়ে দেখ্‌ছে না তা – নাইবা যদি দেখে শীতল মিঠা বাদল ... (Transliteration = Gabhīra kālō mēghēra parē ranina dhanu bām̐kā raṅēra tuli buliẏē mēghē khilāna yēna ām̐kā gabuja ghāsē rōdēra pāśē ālōra kērāmati raṅin bēśē raṅin phulē raṅin prajāpati andha mēẏē dēkhchē nā tā – nā'ibā yadi dēkhē śītala miṭhā bādala ...) (Translation = After the deep black clouds, the colored bows turn the curved brush and arch through the clouds as if the blind girl is not seeing the colorful butterflies in the colorful flowers in the colorful sunshine beside the sun on the painted grass - not if she sees the cool sweet rain ...)	Sukumar Ray (0.49)	Rabindranath Tagore (0.21)	Jibanananda Das (0.35)	Bankim Chandra Chatterjee (0.13)	Sukumar Ray	Yes
Corpus : LD, Type : Bunch of stories Input2 = এতদিন পরে তাহার বসন্তকালের লট্‌কানে রঙের শাড়ি এবং খোঁপার বেলফুলের মালা লজ্জায় বাল্মীকির প্রবেশ কিন্তু বস্তুত তার প্রতি অত্যাচার কোন্‌টা হবে যদি তার কাজ বন্ধ করে দিই কারণ ... (Transliteration = Ētadina parē tāhāra basantakālēra laṭkānē raṅēra śāṛi ēbaṁ khōm̐pāra bēlaphulēra mālā lajjāẏa bālmīkira prabēśa kintu bastuta tāra prati atyācāra kōṇṭā habē yadi tāra kāja bandha karē di'i kāraṇa ...) (Translation = After all this time, Balmiki enters her spring lattice with a sari of colored sari and a garland of bell flowers, but in reality, what will happen to her if we stop her work because ...)	Sarat Chandra Chattopadhyay (0.56)	Sarat Chandra Chattopadhyay (0.23)	Bankim Chandra Chatterjee (0.14)	Bankim Chandra Chatterjee (0.32)	Rabindranath Tagore	No
Corpus : BAAD16, Type : Bunch of stories Input3 = ভালোবাসাকে অবমাননা করে সে-ও জীবনে আর ভালোবাসা পায় না, তখন তার জীবন বড়ো দুর্বিষহ হয়ে পড়ে, বিষিয়ে ওঠে! তখন হয়তো তার বেশি করে তাকেই মনে পড়ে... (Transliteration = Bhālōbāsākē abamānanā karē sē-ō jībanē āra bhālōbāsā pāẏa nā, takhana tāra jībana baṛō durbiṣaha haẏē paṛē, biṣiẏē ōṭhē! Takhana haẏatō tāra bēśi karē tākē'i manē paṛē ...) (Translation = Insulting love, he also does not get love in life, then his life becomes a big misfortune, poison! Then maybe he remembers her more ...)	Rabindranath Tagore (0.47)	Bankim Chandra Chatterjee (0.31)	Humayun Ahmed (0.52)	Taslima Nasrin (0.12)	Kazi Nazrul Islam	No
Corpus : LD, Type : Bankim Essays Input4 = সহস্র বৎসর কাল বৌদ্ধধর্ম্ম ভারতবর্ষের প্রধান ধর্ম্ম ছিল ভারতবর্ষের পুরাবৃত্ত মধ্যে যে সময়টি... (Transliteration = Sahasra baṭsara kāla baud'dhadharm'ma bhāratabarṣēra pradhāna dharm'ma chila bhāratabarṣēra purābṛtta madhyē yē samaẏaṭi...) (Translation = For thousands of years, Buddhism has been the main religion in India ...)	Bankim Chandra Chatterjee (0.59)	Bankim Chandra Chatterjee (0.52)	Rabindranath Tagore (0.02)	Sarat Chandra Chattopadh yay (0.01)	Bankim Chandra Chatterjee	Yes

Figure 6.14: Sample input/output with model performance

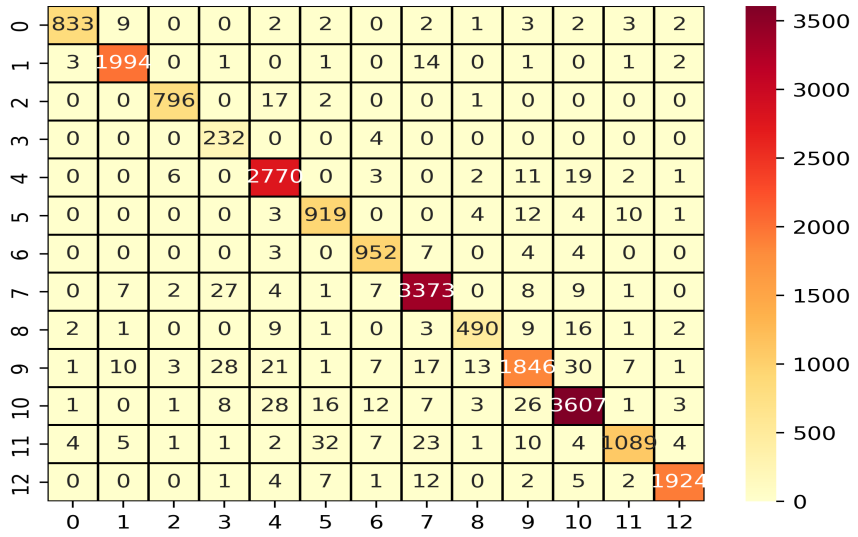
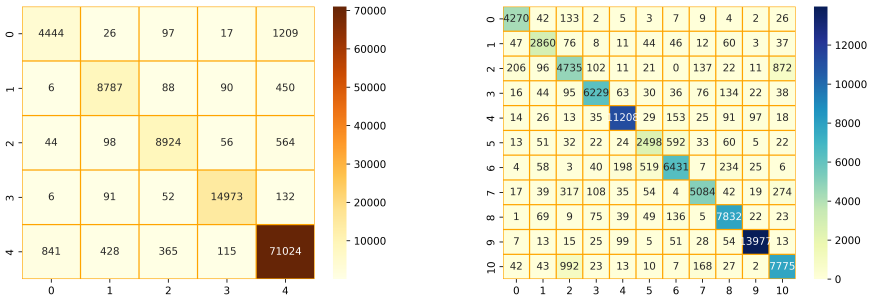


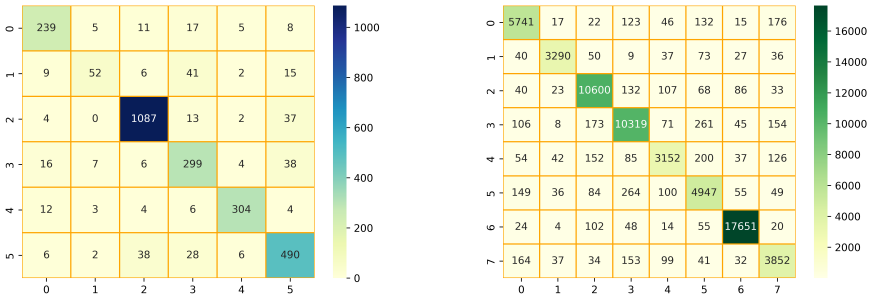
Figure 6.15: Confusion Matrix of BDTCC



(a) BARD Confusion Matrix **(b)** BTCC11 Confusion Matrix

Figure 6.16: Confusion matrix visualization for AVG-M+CNN model with BARD & BTCC11 corpora. BARD categories with index: economy(0), entertainment(1), international(2), sports(3) & state(4). BTCC11 categories with index: Accident(0), Climate-environment(1), Crime(2), Education(3), Entertainment(4), Health(5), Lifestyle(6), Politics(7), Science-Tech(8), Sports(9) & law-justice(10)

minimum error rate (e.g., 5%) has been taken from class index two because of the distinguishable linguistics feature. Fig. 6.17, (b) shows the error and correct prediction of corpus Prothom-Alo. The minimum error (e.g., 2.0%) has been obtained from class index six, whereas the maximum error (e.g., 16.0%) occurred for class index four due to overlapping feature distribution among the other categories.



(a) IndicNLP Confusion Matrix **(b)** Prothom-Alo Confusion Matrix

Figure 6.17: Confusion matrix visualization for AVG-M+CNN models using IndicNLP & Prothom-Alo corpora. IndicNLP categories with index: entertainment(0), international(1), kolkata(2), national(3), sports(4) & state(5). Prothom-Alo categories with index: economy(0), education(1), entertainment(2), international(3), life-style(4), opinion(5), sports(6), technology(7)

The confusion matrix showed that the Sports class achieved the highest accuracy for corpora BARD, Prothom-Alo, and BTCC11, whereas minimum accuracy was obtained for the Crime class (BTCC11) and International (IndicNLP). Due

to category name selection, e.g., corpus IndicNLP International, Kolkata and National are more confusing categories, and the common words are semantically and syntactically different in these classes. However, some words remain in-between categories, e.g., Low-justices and Crime categories of corpus BTCC11. The proposed system did not work well in these cases.

The extrinsic evaluations (e.g., text classification performance) showed that the maximum error rate is 12% for corpus IndicNLP, and the minimum error rate is 4% for corpus IndicNLP. The proposed system showed an error of 10% on the developed corpus (BTCC11). Therefore, the proposed (e.g., AVG-M+CNN) and baseline methods revealed some failure to classify a text accurately. Fig. 6.18 illustrates a few samples of failure cases.

S#.	Copus Code	Input: Translate	Actual Labelled	Correctly Predicted Models	Wrongly Predicted Models	Domain (URL)
1.	C_{f4}	ময়মনসিংহের ... দুর্ঘটনার সত্যতা: Mymensingh ... the truth of the accident	Accident	AVG-M+CNN, GloVe+VDCNN	M-BERT, DistilBERT, IndicBERT, LSTM, GRU, SGD, LibSVM	Newspaper (https://tinyurl.com/kzdvf54w)
2.	C_{f1}	নিউজিল্যান্ড ... সিরিজে টেস্ট : Tests in New Zealand... Series	Sports	Non of Them	All Classification models	Newspaper ((https://tinyurl.com/wkfuspt2)
3.	C_{f3}	বিত্তশালী ... গবেষণা প্রকাশিত হয়েছ: Wealthy ... research published	Technology	AVG-M+CNN	M-BERT, Distil-BERT, Indic-BERT, VDCNN, LSTM, GRU, SGD, LibSVM	Newspaper (https://tinyurl.com/5yarm8br)

Figure 6.18: Few examples of failure text classification

Sample 1 ($S\#$ 1) is correctly predicted by the proposed method (AVG-M+CNN), and GloVe+VDCNN [2]. However, the rest of the baseline models cannot predict correctly due to overlapping mutual information. The second sample ($S\#$ 2) is a Sports news category, but this sports event was held in New Zealand, and the local newspaper agency focused more on international news than sports news. This sample is collected from the BARD corpus, and in this corpus, lots of international news is available in training sets like this pattern. However, more feature distributions come from International categories concerning the Sports category. As a result, all methods cannot provide the correct prediction due to the mutual information between the Sports and International categories. For the third sample ($S\#$ 3), the proposed system correctly predicted it as Technology, whereas the other model cannot predict the actual category. It is confirmed that the proposed and baseline models revealed some limitations when the mutual information existed in a single text.

(iv) Error Analysis of Attention-based Meta-Embedding with ACNN Model: The error analysis provides in-depth insights into the proposed model's performance regarding qualitative and quantitative strengths and weaknesses. Fig. 6.19 shows a quantitative analysis of the CovTiNet system using the confusion matrix.

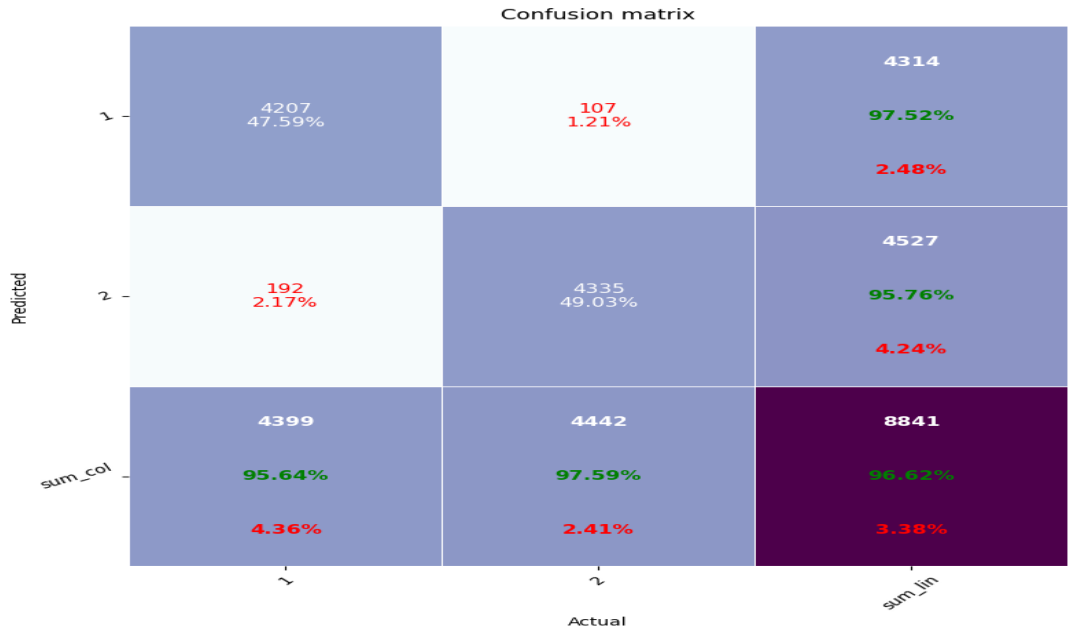


Figure 6.19: Confusion matrix of the proposed model (CovTiNet) on of test samples

A total of 107 out of 4,314 misidentifications occurred in the Covid test samples, whereas 192 out of 4,527 misidentifications occurred in the non-Covid test samples due to joint feature distribution presented in both categories. For example, *Accident* and *Health*-related samples of non-Covid categories contain death-related frequent words, also available in the Covid test samples. As a result, the standard typical word distribution obtained some extra attention, and the model failed to detect the actual category. Overall, a 2.41% error was obtained from the non-Covid category, whereas a 4.36% error occurred in the Covid class with an average error of 3.38%.

Fig. 6.20 shows some test set samples with the actual and predicted labels. The first two Covid test samples (# 1 and #2) are taken from the Newspaper domain. The CovTiNet and ELECTRA-Bengali models correctly predicted the S# 1 text sample. In contrast, the other baseline methods failed to predict the correct labels due to the limitations of feature extraction methods (e.g., shortage of word semantics and context information). The proposed and baseline models cannot predict sample # 2 text samples owing to a shortage of aspect information

(e.g., Covid-related word and semantic information).

S#.	Input: Translate	Actual Labelled	Correctly Predicted Models	Wrongly Predicted Models	Domain (URL)
1.	ক্ষুধার্ত মানুষের ... মোনেম লিমিটেড: Monem Limited ... for hungry people	Covid	CovTiNet & ELECTRA-Bengali	M-BERT, DistilBERT, IndicBERT, BiLSTM, CNN, VDCNN & DCNN	Newspaper (/https://tinyurl.com/2y6puj8v)
2.	কৃষক খামারিদের ... এখনই প্রণোদনা: Incentives for ... farmers now	Covid	Non of Them	All baselines & CovTiNet	Newspaper (https://tinyurl.com/yv2j7k2r)
3.	কুবির উন্নয়নের ... লক্ষে চীনা: Chinese for the ...development of Kubir	Non-Covid	Non of Them	All baselines & CovTiNet	Social Media (https://tinyurl.com/5yarm8br)
4.	শিশুর কিডনিতে ... সার্জারি প্রয়োজন: The baby's kidney ... needs surgery	Non-Covid	CovTiNet	ELECTRA-Bengali, M-BERT, DistilBERT, IndicBERT, BiLSTM, CNN, VDCNN & DCNN	Newspaper (https://tinyurl.com/2c6vvv3n)

Figure 6.20: Actual and predicted test samples

In Fig. 6.20, the third and fourth Non-Covid samples are taken from social media and newspapers, respectively. The baseline and proposed systems do not correctly detect the third sample (i.e., #3) because a large number of words are semantically and syntactically similar to the Covid category [174, 175] whereas the context information is not similar to Covid category. Thus, the proposed (CovTiNet) and baseline methods cannot capture the context information correctly. The proposed model can successfully detect sample #4 text samples that express non-Covid health text samples. The proposed system correctly predicts this sample, but baseline methods failed to detect it. In this sample (#4), most of the words are related to the health category and, like with Covid category words, but the aspect is different (i.e., non-Covid). The proposed system position encoding and attention-based fusion properly extract the semantic, syntactic, and context information, whereas the other methods do not adequately extract that information. As a result, the proposed CovTiNet is better for semantic, syntactic, and aspect-based information retrieval purposes.

6.5.11.1 Error Analysis of Transformer-based Models

The transformer-based language models error analysis are employed in three different tasks, i.e., (i) Arabic Covid Text Identification Model, (ii) Authorship Attribution Model, and (iii) Proposed AFuNet Model. The following paragraphs describe the errors of each model.

(i) Error Analysis of Arabic Covid Text Identification Model: This section describes the proposed system's strengths and weaknesses concerning the error analysis with the confusion matrix. The confusion matrix involves analyzing

the errors that can occur in the AraCovTexFinder model. Fig. 6.21 shows the confusion matrix of the AraCovTexFinder model.

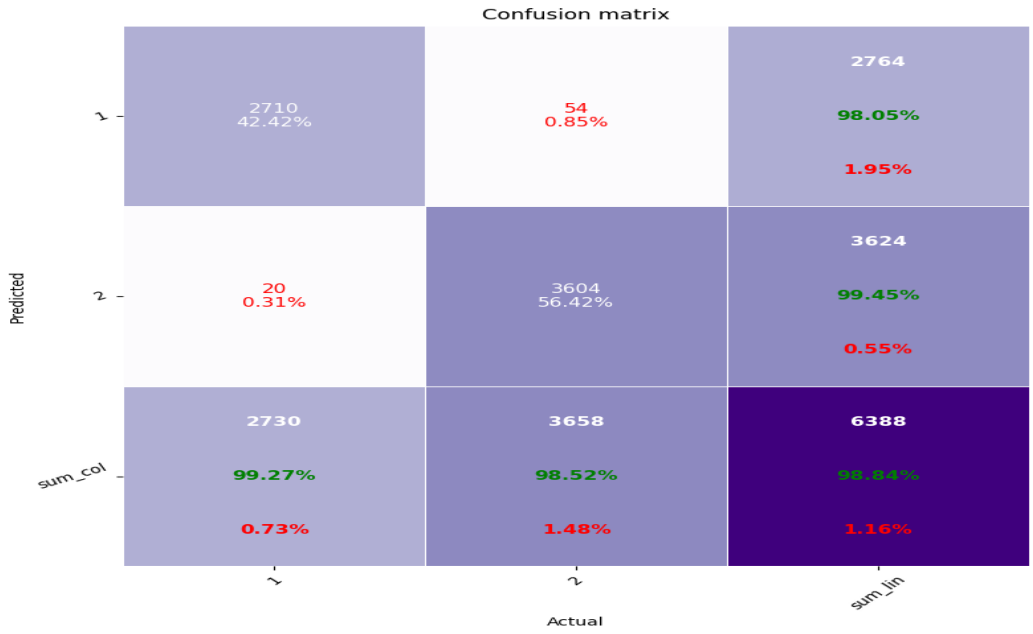


Figure 6.21: Confusion matrix of AraCovTexFinder on test samples

Figure 6.21 shows that the model can correctly predict 2,710 Covid texts, whereas 20 Non-covid texts are incorrectly predicted. On the other hand, out of 2764 COVID-19 texts, 54 are incorrectly predicted where texts did not have COVID information. There are 3,604 non-COVID texts where the model correctly predicted that these text samples do not hold COVID information. The above results indicate that the model has a high accuracy, precision, recall, and F1 score. It correctly detects COVID-19 texts in 98% of the samples. However, it is worth noting that the model's performance is affected by the imbalance in the dataset. The number of non-covid samples is much higher than that of Covid text samples; thus, the model may be biased toward negative cases.

Figure 6.22 shows the proposed system's (i.e., AraCovTexFinder) limitations and strength compared to baseline methods. The proposed system (i.e., S# 1) correctly predicts the first example (i.e., AraCovTexFinder), whereas the other methods failed to predict correctly. The second sample (i.e., S# 2) is a short text of a Covid-related Newspaper headline that failed to predict correctly by the proposed and baseline systems due to a shortage of Covid-related words semantics. Similarly, the third example (i.e., S# 3) is a non-Covid-related short text containing some health-related words. As a result, all methods matched some semantic distribution with the Covid category and failed to predict the actual label (i.e., non-Covid). The AraCovTexFinder, AraBERT, and BERT-

Arabic correctly predicted the fourth example (i.e., S# 4). That means all Arabic language models can accurately predict the fourth example, whereas the multi-lingual models cannot. Figure 6.22 revealed that the classification performance of Arabic language-based transformer models is better than the multi-lingual ones.

S#.	Input: Translate	Ground Truth	Correctly Predicted Models	Wrongly Predicted Models	URL
1.	تونس روبترز قال العضو العلمية باللجنة واقسام تونس في : Tunisia Reuters said the member of the scientific committee in Tunisia and sections	**Covid**	AraCovTexFinder	AraBET, BERT-Arabic, mBERT, XML-RoBERTa, mDeBERTa-V3, mDistilBRT, BiLSTM, CNN, LSTM & DCNN	https://tinyurl.com/26pw9pxr
2.	مجدداً الأمريكي الدفاع وزير يهاجم كورونا: Corona attacks the US Secretary of Defense again	**Covid**	Non of Them	All baselines & AraCovTexFinder	https://tinyurl.com/3wt9fk8z
3.	أكبر تنظم الصحية والعلوم للطب الخيمة رأس للجامعات رياضية بطولة : Ras Al Khaimah for Medicine and Health Sciences organizes the largest sports tournament for universities	**Non-Covid**	Non of Them	All baselines & AraCovTexFinder	https://tinyurl.com/5n7467zm
4.	يبحثان القوى واتحاد الرياضي دبي ,مجلس الاستراتيجية اللعبة خطة : Dubai Sports Council and the Federation of Athletics discuss the game's strategic plan	**Non-Covid**	AraCovTexFinder, AraBERT & BERT-Arabic	mBERT, XML-RoBERTa, mDeBERTa-V3, mDistilBRT, BiLSTM, CNN, LSTM & DCNN	https://tinyurl.com/5bsbxudn

Figure 6.22: Actual and predicted test samples

Although the Arabic text classification systems (i.e., AraCovTexFinder) have achieved reasonable accuracy, they face several limitations. The AraCovTexFinder One major limitation is the limited availability of labeled COVID-19 affiliated data and the failure to predict the short Arabic texts. The COVID-19 pandemic is a relatively recent phenomenon, and the amount of Arabic text data related to the pandemic is still relatively small. This lack of data makes it challenging to train accurate classifiers. Additionally, Arabic script is written right-to-left and has a complex morphology, with words changing form based on context. This constraint can make it difficult for text classification systems. However, the accuracy of these systems still needs to be comparable to that of systems developed for languages with more data and less morphological complexity.

(ii) Error Analysis of Authorship Attribution Model: The error analysis presented in this study provides a comprehensive examination of both the limitations and strengths of the proposed AuthorNet model and the baseline models. Figure 6.23 offers a visual representation of three author text samples from three different corpora, showcasing the ground truth (i.e., the actual author name), accurately attributed models, and models that failed in attribution (i.e., wrongly attributed models).

From Figure 6.23, we can observe three samples: the first sample (S#1)

S#.	Input: Transliteration	Ground Truth	Correctly Attributed Models	Wrongly Attributed Models	Corpus:URL
1.	শ্রাবণের ... সেই মুখের ভিতর প্রবেশ করলাম: Śrābaṇēra... Sē'i mukhēra bhitara prabēśa karalāma	**Jibanananda Das**	Non of Them	All Baseline Models & AuthorNet	BACC-18: https://tinyurl.com/bddcjw3d
2.	তাহলে ... টেলিগ্রাম করে আমাকে জানাবে তোমার চিঠির: Tāhalē... Ṭēligrāma karē āmākē jānābē tōmāra ciṭhira	**Shirshendu Mukhopadhyay**	AuthorNet	All Baselines Models	BADD16: https://tinyurl.com/3km7xafy
3.	ভালো ... বইটির সাহিত্যমানই আমাকে এই বিশ্বাস দান করেছে: Bhālō ... Ba'iṭira sāhityamāna'i āmākē ē'i biśbāsa dāna karēchē	**RN**	Non of Them	All Baselines & AuthorNet	BADD6: https://tinyurl.com/33brm62a

Figure 6.23: Few author text samples with attributed summary

is taken from the BACC-18 corpus, which consists of a poem by "Jibanananda Das." The failure to capture the complete text in the input sample due to sentence segmentation impacts the ability of both the proposed model and the baseline models to capture the actual stylometric patterns.

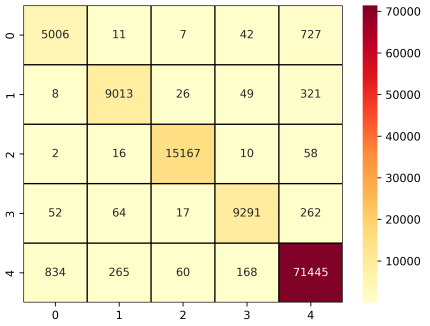
The second sample (S#2) is extracted from the BAAD16 corpus, featuring a story by "Shirshendu Mukhopadhyay." Although the author's text comprises 750 words, the proposed model only considers the first 400 words, resulting in a correct prediction. Conversely, the baseline models fail to provide accurate predictions. This improvement is attributed to the impact of attention-based early fusion operations, which enhance the representation of stylometric features. The third sample (S#3) is derived from the BAAD6 corpus, containing an anonymous blog post by "RN." Due to diverse author text feature distribution and morphological variations, such as a mixture of Cholito Bhasha (colloquial language) and transliteration, the proposed AuthorNet model and all of the baseline models struggle to make correct predictions.

In conclusion, the error analysis reveals that the proposed AuthorNet model has certain limitations. It faces challenges in providing accurate predictions for sentence-level augmented texts and samples containing morphologically mixed words (including Cholito Bhasha, Sadhu Bhasha, and transliteration). However, when compared to the baseline models, the proposed AuthorNet model demonstrates a lower error rate, indicating its superior performance.

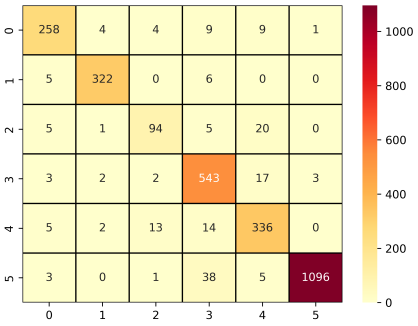
(iii) Error Analysis of Proposed AFuNeT Model: The error analysis delves into the intricacies of the proposed system, AFuNet, highlighting its strengths and weaknesses compared to previous Bengali text classification models. The error analysis of AFuNet was performed in quantitative and qualitative ways.

6.5.12 Quantitative Analysis

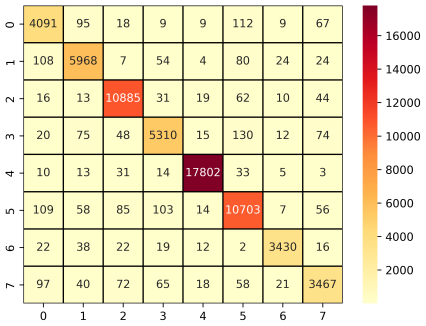
Confusion matrices are used to perform the quantitative error analysis of AFuNet. Figure 6.24 depicts the confusion matrices with heatmaps of the proposed model regarding BARD, IndicNLP, and Prothom-Alo datasets.



(a) Confusion Matrix of BARD



(b) Confusion Matrix of IndicNLP



(c) Confusion Matrix of Prothom-Alo

Figure 6.24: confusion matrices of AFuNet model for the datasets: BARD, IndicNLP, and Prothom-Alo. BARD categories denoted by the indexes: economy(0), entertainment(1), international(2), sports(3) & state(4); IndicNLP categories represent with indexes: entertainment(0), international(1), Kolkata(2), national(3), sports(4) and state(5); Prothom-Alo categories indicates with indexes: economy(0), education(1), entertainment(2), international(3), lifestyle(4), opinion(5), sports(6), and technology(7)

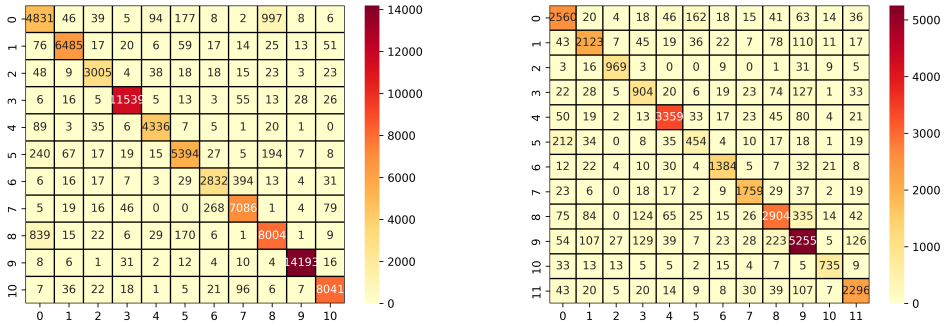
Figure 6.24a represents the confusion matrix of AFuNet for the BARD corpus. Notably, the maximum error, totaling 787 misclassifications out of 5,793 instances, occurred at index 0 (*economy* category). On the other hand, the minimum error, involving 86 misclassifications out of 15,253 cases, was observed at index 2 (*international* category). This discrepancy in error rates suggests that the *economy* category faces challenges in achieving better accuracy due to class imbalance issues and limited training data distribution.

Figure 6.24b shows the confusion matrix of AFuNet for the IndicNLP corpus.

The highest number of misclassifications, totaling 31 errors out of 125 instances, occurred at index 2 (corresponding to the *Kolkata* category). This performance can be attributed to fewer training samples and issues related to within-class imbalance.

Continuing with Figure 6.24c for the Prothom-Alo dataset, the maximum misclassifications, amounting to 371 errors out of 3,838 instances, were observed at index 7 (representing the *technology* category). This misclassification pattern is influenced by the mutual distribution of data points with index 0 and 2. Conversely, the minimum misclassification, involving 109 errors out of 17,911 instances, was recorded at index 4 (about the *lifestyle* category). This outcome can be attributed to an adequate number of category samples and a diverse class distribution within this category.

Figure 6.25 demonstrates the confusion matrices of AFuNet concerning BTCC11 and OSBC datasets.



(a) Confusion Matrix of BTCC11 **(b)** Confusion Matrix of OSBC

Figure 6.25: Confusion matrices of AFuNet model concerning BTCC11 and OSBC corpora. The classes of BTCC11 dataset represents with various indexes: Accident(0), Climate-environment(1), Crime(2), Education(3), Entertainment(4), Health(5), Lifestyle(6), Politics(7), Science-Tech(8), Sports(9) and law-justice(10). The classes of the OSBC dataset are denoted with various indexes: Entertainment (0), Opinion (1), Crime (2), Economics (3), Sports (4), Art (5), Environment (6), International (7), Education (8), Politics (9), Science_tech (10), Accident (11)

Figure 6.25a shows the confusion matrix of AFuNet for the BTCC11 dataset. The maximum misclassifications, totaling 1,382 out of 6,213 samples, occurred at index 0 (*Accident* category). This high misclassification rate can be attributed to Out-of-Distribution (OOD) issues and relatively fewer training samples than other categories within the corpus. Conversely, the minimum number of misclassifications, involving 94 errors out of 14,287 pieces, was observed at index 9 (*Sports*). This can occur in the category with the most significant number of

training samples and a diverse data distribution compared to other categories within the corpus.

Figure 6.25b illustrates the confusion matrix of the proposed model for the OSBC corpus. The highest classification errors were observed at index 5, corresponding to the *Art* category, while index 4 (*Sports*) noted the lowest classification errors. The challenges that the *Art* category faces in achieving better accuracy can be attributed to class imbalance issues within the OSBC corpus.

6.5.13 Qualitative Analysis

Figure 6.26 illustrates the qualitative analysis with three distinct samples. Sample 1 (S#), extracted from the IndicNLP corpus, and Sample 2 (S#), sourced from the Prothom-Alo corpus, were both subjected to evaluation by best-performing eight baseline models as well as the proposed AFuNet. Surprisingly, neither of the samples was capable of making accurate predictions using these employed models. The actual categories for these samples were *Entertainment* for Sample 1 and *Technology* for Sample 2. However, it was evident that these samples exhibited common distributional characteristics with other categories within the corpus. This overlap in distribution, combined with the limited amount of text data in the training set, led to erroneous predictions.

S#.	Corpus	Input: Translate	Actual Labelled	Correctly Predicted Models	Wrongly Predicted Models	Domain (URL)
1.	IndicNLP	রাতারাতি ইন্টারনেট ... সত্যতা ছবিটি: Overnight internet ... the truth picture	Entertainment	Non of Them	All Classification Models	Newspaper (https://tinyurl.com/4yae2y4r)
2.	ProthomAlo	ডিম্বাশয়ের টিস্যু ... খুব কঠিন : Ovarian tissue is ... very hard	Technology	Non of Them	All Classification Models	Newspaper (https://tinyurl.com/3pzfpydn)
3.	BTCC11	আজহারুলের মৃত্যুদণ্ড ... ঘোষণা করলেন: Azharul's death sentence... declared:	Crime	FusionNetTC	All Baseline Models	Newspaper (https://tinyurl.com/24h7nmdf)

Figure 6.26: Examples of text classification errors

Sample 3 (S#), sourced from the BTCC1 corpus, posed a unique challenge for evaluation. Interestingly, it was correctly classified by the proposed AFuNet, but it confounded all the best-performing baseline models, resulting in incorrect predictions. The actual category of Sample 3 was *Crime*, yet more than 60% of its words overlapped with the "Law-Justice" category. This shared set of words across both types presented a significant obstacle for the baseline models, causing

them to make erroneous predictions. However, utilizing attention operations and early fusion within the AFuNet allowed for the discrimination of common words from category-specific ones, ultimately leading to the correct prediction.

6.5.14 Impact of Text Classification in Multimodal Dataset

Multimodal text classification involves classifying textual data, which is achieved by considering both textual and visual content. The individual classification of textual and visual content is performed separately, and the final class label is generated by combining the semantics from both modalities. This approach leverages the complementary information provided by text and visual elements to enhance the overall classification accuracy and capture a more comprehensive understanding of the content.

In this study, we explored Bengali multimodal text classification, dealing with scenarios where textual content is limited, specifically very short text, and supplemented with available image data. To facilitate this investigation, we constructed a Bengali Multimodal Text Classification Corpus (BMMTC), detailed in Table 3.12. The research utilized a multimodal text classification system, and the performance outcomes are outlined in Table 6.50.

Table 6.50: Summary of multimodal text classification on BMMTC corpus

Modality	Model	A(%)	No. of Test Samples
Visual	ResNet18	87.66±0.2	3000
	ResNet34	85.23±0.2	
	ResNet50	79.48±0.2	
	AlexNet	75.38±0.2	
	VGG16	86.59±0.2	
	VGG19	83.70±0.2	
	ViT	**90.13±0.2**	
Textual	CNN	85.93±0.2	3000
	BiLSTM	86.66±0.2	
	BanglaBERT	**91.78±0.2**	
	MuRIL	89.67±0.2	
Visual+Textual	**BanglaBERT+ViT**	**92.68±0.2**	**3000**
	BanglaBERT+ResNet18	90.74±0.2	
	MURIL+ViT	90.97±0.2	
	MuRIL+ResNet18	88.47±0.2	

Table 6.50 summarizes the impact of multimodal text classification on Bengali text classification corpus (i.e., BMMTC), considering various modalities and models. In the visual modality, ResNet18, ResNet34, ResNet50, AlexNet, VGG16,

VGG19, and ViT models were employed, with ViT achieving the highest accuracy of 90.13±0.2%. For the textual modality, CNN, BiLSTM, BanglaBERT, and MuRIL models were utilized, and BanglaBERT achieved the highest accuracy of 91.78±0.2%. In the combined Visual+Textual modality, different combinations such as BanglaBERT+ViT, BanglaBERT+ResNet18, MURIL+ViT, and MuRIL+ResNet18 were tested, with BanglaBERT+ViT demonstrating the highest accuracy of 92.68±0.2%. The observed accuracy improvement in multimodal text classification is attributed to using complementary information from both textual and visual modalities. Models that combine representations from BanglaBERT and ViT exhibit enhanced performance by leveraging the robustness and semantic correlation offered by each modality. This fusion of diverse features leads to improved generalization, a reduction in modality-specific limitations, and effective handling of complex relationships within the data, resulting in a more comprehensive understanding of content and surpassing the performance of models relying solely on individual modalities.

6.5.15 Discussion

This research developed an *intelligent* text classification system, AFuNet, tailored for low-resource scenarios and evaluated using *five Bengali text classification corpora*. Throughout the development of AFuNet, this study illuminates key characteristics and unveils findings as follows.

- This study utilizes two distinct categories of text-to-feature extraction models: non-contextual models (comprising GloVe, Word2Vec, and FastText) and contextual models (including BanglaBERT, BELECTRA, MuRIL, DeBERTa, MBERT, XLMRoBerta, IndicBERT, and DistilBERT). Within the contextual category, these models are further classified into multilingual models (MuRIL, DeBERTa, MBERT, XLMRoBerta, IndicBERT, and DistilBERT) and monolingual models (specifically, BanglaBERT and BELECTRA). While both BanglaBERT and BELECTRA, as monolingual language models, are trained using Bengali corpora with ELECTRA models, BELECTRA faces challenges in achieving lower accuracy primarily due to vocabulary limitations compared to BanglaBERT. On a broader scale, the BanglaBERT model outperforms other language models by effectively addressing Out-of-Vocabulary (OOV) word issues and proficiently extracting context-aware embedding features (Details in Table 6.38).

- Fine-tuned the hyperparameters for low-resource Bengali text classification tasks and conducted massive experiments on five Bengali text classification

corpora. In the context of Bengali text classification, the results presented in Table 6.32 and Table 6.33 indicate that the hyperparameter tuning process, involving batch size, maximum sequence length, and number of epochs adjustments, has yielded the most favorable gradient for backpropagation phase.

- Attention-based early fusion significantly enhances classification accuracy for low-resource languages, including Bengali. With its morphological richness and linguistic variations, such as Sadhu-bhasha, Cholito-bhasha, and Sanskrit-bhasha, Bengali presents a unique challenge. Due to this linguistic diversity, single transformer-based models often struggle to achieve superior accuracy compared to fusion-based models. However, in the case of AFuNet, an attention-based early fusion approach is employed. This approach combines the strengths of the best-performing monolingual transformer model (BanglaBERT) with the best-performing multilingual model (MuRIL). Together, their embeddings adeptly capture linguistic variations, improving Bengali text classification accuracy. Furthermore, the incorporation of focal loss effectively addresses class imbalance issues, further enhancing the model's overall accuracy.

- Among the five corpora, the BARD corpus contained the highest number of text samples, i.e., 376,205, and holds the minimum number of categories (i.e., 5). Conversely, the OSBC corpus has the maximum number of categories, i.e., 12, but contains relatively few text samples (i.e., 95,855). This disparity in data distribution contributes to the BARD corpus achieving an impressive maximum accuracy of 97.35%. while the OSBC corpus records a comparatively lower accuracy of 85.37% *Overall, the performance of the AFuNet for the Bengali text classification system is heavily influenced by several key factors. These include the size of vocabularies within language models, the two highly influential hyperparameters: batch size and maximum sequence length, the utilization of attention-based fusion operations, and the implementation of focal loss to address class imbalance within the corpus.*

- **Advantages of semi-automated annotation system:** Semi-automated annotation systems significantly reduce the manual effort required for annotation by automating certain aspects of the process. This leads to faster annotation times and increased productivity. Automated components of the system ensure consistent annotation standards across large datasets, minimizing variability introduced by human annotators. By reducing the

reliance on manual labor, semi-automated annotation systems can lower overall annotation costs, particularly for large-scale projects. These systems are highly scalable and can handle large volumes of data efficiently, making them suitable for projects with extensive annotation requirements.

- **Disadvantages of semi-automated annotation system:** Despite automation, semi-automated systems are susceptible to errors, which can propagate throughout the annotation process and affect the quality of labeled data. Implementing and maintaining a semi-automated annotation system can be complex and resource-intensive, requiring expertise in both annotation methods and technology. Some types of data or annotation tasks may not be well-suited for semi-automated approaches, particularly those requiring nuanced understanding or subjective judgment. Semi-automated systems rely heavily on the performance and reliability of the underlying technology, which may introduce vulnerabilities or limitations.

6.6 Summary

Chapter 6 commences by delineating three types of non-contextual embedding models: Individual embedding, Meta-Embedding, and Attention-based Meta-Embedding. Through intrinsic evaluations, three best-performing individual embeddings (GloVe, FastText, and Word2Vec), one Meta-embedding (AVG-M), and one Attention-based Meta-Embedding (APeAGF) are selected for integration into text classification models. The proposed attention-based early fusion model of transformer-based language (AFuNeT) showcases superior performance, substantiated by evaluations across multiple authorship attribution and document classification corpora. The chapter further presents error cases for the proposed system and baseline models, accompanied by illustrative examples. Finally, the discussion section offers valuable insights into the AFuNeT model.

Chapter 7

Conclusion and Future Works

This study has introduced a text classification system for resource-constrained environments, demonstrating its ability to intelligently classify Bengali text. Within this chapter, we will begin by summarizing potential responses to the research questions, drawing from the significant contributions made in this study. Subsequently, we will detail the constraints and implications of the proposed system. To address these limitations, we present a set of future research recommendations to enhance the resource-constrained text classification system we have developed. Lastly, we furnish a bibliography of publications.

The answers to the research questions, derived from the significant contributions, are as follows:

- This research has developed an active learning-based system for resource-constrained corpus development and creating a Bengali text classification corpus named BTCC11 (referred to as **ARQ1** and detailed in Section 3.3). By employing manual annotation and verification techniques, this study has also developed five text classification corpora: BACC-18, BDTCC, AraCoV, BCovC, and ECoVC. In addition, five unlabeled embedding corpora have been developed, including EC, WEC, CovEC, ECoVE, and AraEC.

- By utilizing the transformer-based language models, this research effectively overcomes the problem of Out-of-Vocabulary (OOV) words, which is demonstrated in Section 5.4.3.6 (i.e., **ARQ2**). This approach significantly enhances the performance of Bengali text classification, as evidenced in Table 6.22, Table 6.30, Table 6.46, and Table 6.38.

- In this study, two hyperparameter tuning algorithms have been developed. The first algorithm is designed for deep learning-based models (detailed in Section 5.3.1, Algorithm 9), while the second algorithm is tailored for

transformer-based language models (outlined in Section 5.4.1, Algorithm 10) (i.e., **ARQ3**). The comprehensive performance of both the tuned and initial hyperparameters is summarized in Figures 6.8 and 6.6, as well as Table 6.29.

- An attention-based early fusion resource-constrained text classification system has been developed, which involved the statistical significance test-based selection of the top-performing two transformer-based models and their fusion (as outlined in **ARQ4** and detailed in Section 5.4.3.5). The accuracy improvements achieved by the proposed AFuNeTs are summarized in Table 6.35, Table 6.36, and Figure 6.7.
- The focal loss function was utilized to address and mitigate class-imbalance issues, as illustrated in Table 6.33 (i.e., **ARQ5**).

7.1 Weaknesses of the Proposed Model

While the proposed AFuNeT model has attained state-of-the-art (SOTA) performance, it still exhibits certain limitations. The identified weaknesses include:

- An active learning-based corpora development proposed in this research has successfully reduced manual annotation time and cost, it is worth noting that the system still requires some initial category-based dictionary words, particularly for zero-resource languages.
- Utilized the pre-trained monolingual and multilingual transformer-based language models; however, domain-specific language models were not trained based on the corpora used. The hyperparameters tuning algorithm employed in this study follows a brute-force approach, which can be time and memory-consuming. Our proposed system does not effectively handle out-of-distribution (OOD) instances and uncertainty.
- The proposed AFuNet has demonstrated commendable accuracy across five corpora, but it still has certain limitations. Notably, the focal loss employed in the model falls short of thoroughly addressing class imbalance issues.
- Due to the use of Byte Pair Encoding (BPE) for Bengali Out-of-Vocabulary (OOV) words, the model struggles to capture context-aware word-level semantic and syntactic meanings accurately. There may be better choices than AFuNet for low-resource intrinsic analyses, such as word-level morphological analysis, which includes tasks like measuring cosine similarity

between words, analogy tasks, and conducting word-level grammatical error checks.

7.2 Possible Implications

The study offers two main categories of implications: theoretical and practical. These implications have far-reaching consequences for the field of resource-constrained Bengali text classification. Theoretical implications are centered on advancing the understanding of text classification systems, while practical implications focus on real-world applications and their impact on various stakeholders. Together, these implications contribute to the overall significance of the study.

Theoretical Implications: This study contributes to the theoretical advancements in the domain of resource-constrained Bengali text classification. It introduces a novel approach to corpus development through active learning-based methods, which has the potential to significantly reduce the resource requirements for building effective classification systems. Furthermore, the attention-based Meta-Embedding system provides insights into optimizing feature representations for low-resource languages. The hyperparameters tuning Algorithm offers a systematic approach to enhancing model performance. The utilization of attention-based early fusion with monolingual and multilingual transformer-based language models demonstrates the adaptability of state-of-the-art techniques to resource-constrained languages. Lastly, the incorporation of the focal loss function for addressing class imbalance issues expands the understanding of handling imbalanced datasets in low-resource language scenarios.

Practical Implications: The practical implications of this research are substantial. The active learning-based corpora development system offers a cost-effective and efficient way to build datasets for resource-constrained languages, thus reducing the manual annotation efforts required. The resource-constrained Bengali text classification system developed in this study finds applications in diverse domains. It can be employed for content moderation to filter out inappropriate online content, enhance information retrieval by categorizing web data, perform sentiment analysis for understanding public opinion, automate news categorization, detect and filter spam, verify authorship of text, assist in language translation, improve healthcare record management, enhance e-commerce product recommendations, and facilitate educational assessment. This system's versatility enables its utilization in many industries, offering more efficient and

accurate solutions across various applications.

7.3 Future Recommendations

While the proposed system has demonstrated commendable performance in addressing various challenges, such as mitigating corpora scarcity, resolving embedding and classification model availability concerns, improving hyperparameters tuning, and handling out-of-vocabulary (OOV) words, further enhancements can be achieved by implementing the following recommendations:

- Implement a few-shot with the prompt-based system for corpora development.
- Train large language models (LLMs) using Bengali embedding corpora.
- Apply prompt-based hyperparameter tuning for domain-specific classification models.
- Incorporate Bayesian models to address out-of-distribution (OOD) and uncertainty issues.
- Implement random under-sampling, over-sampling, and synthetic over-sampling, i.e., SMOTE

7.4 Publications

Journals

[**1**] M. R. Hossain, M. M. Hoque, M. A. A. Dewan, N. Siddique, M. N. Islam and I. H. Sarker, "Authorship Classification in a Resource Constraint Language Using Convolutional Neural Networks," in **IEEE Access**, IEEE, vol. 9, pp. 100319-100338, 2021, doi: 10.1109/ACCESS.2021.3095967.

[**2**] M. R. Hossain, M. M. Hoque, N. Siddique, and I. H. Sarker, "Bengali text document categorization based on very deep convolution neural network," in **Expert Systems with Applications**, Elsevier, vol. 184, pp. 115394, 2021, doi: 10.1016/j.eswa.2021.115394.

[**3**] M. R. Hossain, M. M. Hoque, N. Siddique, and I. H. Sarker, "CovTiNet: Covid text identification network using attention-based positional embedding feature fusion," in **Neural Computing and Applications**, Springer Nature, vol. 35, no. 18, pp. 13503-13527, 2023, doi: 10.1007/s00521-023-08442-y.

[**4**] M. R. Hossain, M. M. Hoque, and N. Siddique, "Leveraging the meta-embedding for text classification in a resource-constrained language," in **Engineering Applications of Artificial Intelligence**, Elsevier, vol. 124, pp. 106586, 2023, doi: 10.1016/j.engappai.2023.106586.

[**5**] M. R. Hossain, M. M. Hoque, N. Siddique, and M. A. A. Dewan, "AraCovTexFinder: Leveraging the Transformer-based Language Model for Arabic COVID-19 Text Identification", in **Engineering Applications of Artificial Intelligence**, Elsevier, vol. 133, pp. 107987, 2024, doi: 10.1016/j.engappai.2024.107987.

Conferences

[**1**] M. R. Hossain, and M. M. Hoque, "Towards Bengali Word Embedding: Corpus Creation, Intrinsic and Extrinsic Evaluations," in Proceedings of the 17th **International Conference on Natural Language Processing (ICON)**, 2020, pp. 453–459, url: https://aclanthology.org/2020.icon-main.61

[**2**] M. R. Hossain, and M. M. Hoque, "Semantic Meaning Based Bengali Web Text Categorization Using Deep Convolutional and Recurrent Neural Networks (DCRNNs)," in **Internet of Things and Connected Technologies**, 2021, pp. 494–505, isbn: 978-3-030-76736-5, doi: 10.1007/978-3-030-76736-5_45.

[**3**] M. R. Hossain, M. M. Hoque, and I. H. Sarker, "Text Classification Using Convolution Neural Networks with FastText Embedding," in **Hybrid Intelligent Systems**, 2021, pp. 103–113, isbn: 978-3-030-73050-5, doi: 10.1007/978-3-030-73050-5_11

[**4**] M. R. Hossain, and M. M. Hoque, CovTexMiner: Covid Text Mining Using CNN with Domain-Specific GloVe Embedding," in **Intelligent Computing & Optimization**, 2023, pp. 65–74, isbn: 978-3-031-19958-5, doi: 10.1007/978-3-031-19958-5_7.

[**5**] M. R. Hossain, and M. M. Hoque, "Toward Embedding Hyperparameters Optimization: Analyzing Their Impacts on Deep Leaning-Based Text Classification," in **The Fourth Industrial Revolution and Beyond**, 2023, pp. 501–512, isbn=978-981-19-8032-9, doi: 10.1007/978-981-19-8032-9_35

[**6**] M. R. Hossain, and M. M. Hoque, "CoBertTC: Covid-19 Text Classification using Transformer-based Language Models", in **Intelligent Computing & Optimization, Springer**, 2023 (In Press).

Bibliography

[1] H. Wen, Z. Xiao, E. Hovy, and A. Hauptmann, "Towards open-domain Twitter user profile inference," in *Findings of the Association for Computational Linguistics: ACL 2023*, (Toronto, Canada), pp. 3172–3188, Association for Computational Linguistics, July 2023.

[2] M. R. Hossain, M. M. Hoque, N. Siddique, and I. H. Sarker, "Bengali text document categorization based on very deep convolution neural network," *Expert Systems with Applications*, vol. 184, p. 115394, 2021.

[3] M. R. Hossain, M. M. Hoque, N. Siddique, and I. H. Sarker, "Covtinet: Covid text identification network using attention-based positional embedding feature fusion," *Neural Computing and Applications*, Mar 2023.

[4] A. K. Mandal and R. Sen, "Supervised learning methods for bangla web document categorization," *CoRR*, vol. abs/1410.2045, 2014.

[5] P. Patwa, M. Bhardwaj, V. Guptha, G. Kumari, S. Sharma, S. PYKL, A. Das, A. Ekbal, M. S. Akhtar, T. Chakraborty, K. Shu, H. R. Bernard, and H. Liu, "Overview of constraint 2021 shared tasks: Detecting english covid-19 fake news and hindi hostile posts," in *Combating Online Hostile Posts in Regional Languages during Emergency Situation*, (Cham), pp. 42–53, Springer International Publishing, 2021.

[6] R. Johnson and T. Zhang, "Deep pyramid convolutional neural networks for text categorization," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Vancouver, Canada), pp. 562–570, Association for Computational Linguistics, July 2017.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. ICLR*, (Scottsdale, Arizona, USA), pp. 1–12, 2013.

[8] C. M. Jeffrey Pennington, Richard Socher, "Glove: Global vectors for word representation," in *Proc. EMNLP*, (Doha, Qatar), pp. 1532–1543, ACL, 2014.

[9] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Tran. ACL*, vol. 5, pp. 135–146, June 2017.

[10] Y. Zhang, S. Roller, and B. C. Wallace., "MGNC-CNN: A simple approach to exploiting multiple word embeddings for sentence classification," in *Proc. NAACL*, (San Diego, California), pp. 1522–1527, June 2016.

[11] Y. Huang, H. Peng, Q. Liu, Q. Yang, J. Wang, D. Orellana-Martín, and M. J. Pérez-Jiménez, "Attention-enabled gated spiking neural p model for aspect-level sentiment classification," *Neural Networks*, vol. 157, pp. 437–443, 2023.

[12] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. EMNLP*, (Doha, Qatar), pp. 1746–1751, 2014.

[13] A. Bander, D. Edwin, H. Richard, M. Spiros, and G. Rachel, "Source code authorship attribution using long short-term memory based networks," in *Proc. ESORICS 2017*, (Oslo, Norway), pp. 65–82, 2017.

[14] D. S. Moirangthem and M. Lee, "Hierarchical and lateral multiple timescales gated recurrent units with pre-trained encoder for long text classification," *Expert Systems with Applications*, vol. 165, p. 113898, 2021.

[15] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, (Valencia, Spain), pp. 1107–1116, Association for Computational Linguistics, Apr. 2017.

[16] T. Pires, E. Schlinger, and D. Garrette, "How multilingual is multilingual BERT?," in *Proc. ACL'57*, (Florence, Italy), pp. 4996–5001, ACL, July 2019.

[17] K. Clark, M. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: pre-training text encoders as discriminators rather than generators," *CoRR*, vol. abs/2003.10555, 2020.

[18] P. He, J. Gao, and W. Chen, "Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing," *CoRR*, vol. abs/2111.09543, 2021.

[19] U. Pal, A. S. Nipu, and S. Ismail, "A machine learning approach for stylometric analysis of bangla literature," in *Proc. ICCIT*, (Dhaka, Bangladesh), pp. 1–5, 2017.

[20] T. Chung, B. Xu, Y. Liu, C. Ouyang, S. Li, and L. Luo, "Empirical study on character level neural network classifier for chinese text," *Engineering Applications of Artificial Intelligence*, vol. 80, pp. 1–7, 2019.

[21] B. Catanzaro, N. Sundaram, and K. Keutzer, "Fast support vector machine training and classification on graphics processors," in *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, vol. 307 of *ACM International Conference Proceeding Series*, pp. 104–111, ACM, 2008.

[22] M. R. Hossain and M. M. Hoque, "Automatic bengali document categorization based on word embedding and statistical learning approaches," in *Proc. IC4ME2*, (Rajshahi, Bangladesh), pp. 1–6, 2018.

[23] S. Khanuja, D. Bansal, S. Mehtani, S. Khosla, A. Dey, B. Gopalan, D. K. Margam, P. Aggarwal, R. T. Nagipogu, S. Dave, S. Gupta, S. C. B. Gali, V. Subramanian, and P. P. Talukdar, "Muril: Multilingual representations for indian languages," *CoRR*, vol. abs/2103.10730, 2021.

[24] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," in *Proc. ACL*, (Online), pp. 8440–8451, July 2020.

[25] D. Kakwani, A. Kunchukuttan, S. Golla, G. N.C., A. Bhattacharyya, M. M. Khapra, and P. Kumar, "IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, (Online), pp. 4948–4961, Association for Computational Linguistics, Nov. 2020.

[26] K. Jain, A. Deshpande, K. Shridhar, F. Laumann, and A. Dash, "Indic-transformers: an analysis of transformer language models for indian languages," *CoRR*, vol. abs/2011.02323, November 2020.

[27] C. C. Aggarwal and C. Zhai, *A Survey of Text Classification Algorithms*, pp. 163–222. Boston, MA: Springer US, 2012.

[28] M. R. Hossain, M. M. Hoque, and N. Siddique, "Leveraging the meta-embedding for text classification in a resource-constrained language," *Engineering Applications of Artificial Intelligence*, vol. 124, p. 106586, 2023.

[29] G. Haffari and A. Sarkar, "Active learning for multilingual statistical machine translation," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, (Suntec, Singapore), pp. 181–189, Association for Computational Linguistics, Aug. 2009.

[30] N. S. Dash, *Corpus Text Annotation*, pp. 1–23. Singapore: Springer Singapore, 2021.

[31] C. Sammut and G. I. Webb, eds., *TF–IDF*, pp. 986–987. Boston, MA: Springer US, 2010.

[32] R. S. Bhowmick, I. Ganguli, and J. Sil, "Character-level inclusive transformer architecture for information gain in low resource code-mixed language," *Neural Computing and Applications*, Mar 2022.

[33] M. K. Elhadad, K. F. Li, and F. Gebali, "Detecting misleading information on covid-19," *IEEE Access*, vol. 8, pp. 165201–165215, 2020.

[34] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[35] T. Yu and H. Zhu, "Hyper-parameter optimization: A review of algorithms and applications," *CoRR*, vol. abs/2003.05689, 2020.

[36] A. Khatun, A. Rahman, M. S. Islam, H. A. Chowdhury, and A. Tasnim, "Authorship attribution in bangla literature (aabl) via transfer learning using ulmfit," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, apr 2022.

[37] T. G. Dietterich, "Ensemble methods in machine learning," MCS '00, (Berlin, Heidelberg), p. 1–15, Springer-Verlag, 2000.

[38] U. Arora, W. Huang, and H. He, "Types of out-of-distribution texts and how to detect them," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 10687–10701, Association for Computational Linguistics, Nov. 2021.

[39] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *ICML*, 2011.

[40] E. B. Varghese and S. M. Thampi, "A multimodal deep fusion graph framework to detect social distancing violations and fcgs in pandemic surveillance," *Engineering Applications of Artificial Intelligence*, vol. 103, p. 104305, 2021.

[41] R. M. West, "Best practice in statistics: Use the welch t-test when testing the difference between two groups," *Annals of Clinical Biochemistry*, vol. 58, no. 4, pp. 267–269, 2021. PMID: 33562996.

[42] Y. T. Cao, Y. Pruksachatkun, K.-W. Chang, R. Gupta, V. Kumar, J. Dhamala, and A. Galstyan, "On the intrinsic and extrinsic fairness evaluation metrics for contextualized language representations," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, (Dublin, Ireland), pp. 561–570, Association for Computational Linguistics, May 2022.

[43] I. V. Tetko, D. J. Livingstone, and A. I. Luik, "Neural network studies. 1. comparison of overfitting and overtraining," *J. of Chem. Info. & Com. Sci.*, vol. 35, pp. 826–833, September 1995.

[44] Y. Zhang, C. Yuan, X. Wang, Z. Bai, and Y. Liu, "Learn to adapt for generalized zero-shot text classification," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Dublin, Ireland), pp. 517–527, Association for Computational Linguistics, May 2022.

[45] S. Hochreiter and J. A. Schmidhuber, "Long short-term memory," vol. 9, no. 8, pp. 1735–1780, 1997.

[46] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, (Lisbon, Portugal), pp. 1422–1432, Association for Computational Linguistics, Sept. 2015.

[47] A. F. Bulagang, N. G. Weng, J. Mountstephens, and J. Teo, "A review of recent approaches for emotion classification using electrocardiography and electrodermography signals," *Informatics in Medicine Unlocked*, vol. 20, p. 100363, 2020.

[48] S. Phani, S. Lahiri, and A. Biswas, "A supervised learning approach for authorship attribution of bengali literary texts," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 16, aug 2017.

[49] J. Pavlopoulos, J. Sorensen, L. Dixon, N. Thain, and I. Androutsopoulos, "Toxicity detection: Does context really matter?," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 4296–4305, Association for Computational Linguistics, July 2020.

[50] L. Yao, C. Mao, and Y. Luo, "Clinical text classification with rule-based features and knowledge-guided convolutional neural networks," *BMC Medical Informatics and Decision Making*, vol. 19, p. 71, Apr 2019.

[51] M. Z. Hossain, M. A. Rahman, M. S. Islam, and S. Kar, "BanFakeNews: A dataset for detecting fake news in Bangla," in *Proceedings of the 12th Language Resources and Evaluation Conference*, (Marseille, France), pp. 2862–2871, European Language Resources Association, May 2020.

[52] R. Rojowiec, B. Roth, and M. Fink, "Intent recognition in doctor-patient interviews," in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, (Marseille, France), pp. 702–709, European Language Resources Association, May 2020.

[53] G. Goldin, E. Rabinovich, and S. Wintner, "Native language identification with user generated content," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, (Brussels, Belgium), pp. 3591–3601, Association for Computational Linguistics, Oct.-Nov. 2018.

[54] S.-S. Kang, "Keyword-based document clustering," in *Proceedings of the Sixth International Workshop on Information Retrieval with Asian Languages*, (Sapporo, Japan), pp. 132–137, Association for Computational Linguistics, July 2003.

[55] A. Ekbal, R. Haque, and S. Bandyopadhyay, "Named entity recognition in Bengali: A conditional random field approach," in *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*, 2008.

[56] A. Mukherjee and B. Liu, "Improving gender classification of blog authors," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, (Cambridge, MA), pp. 207–217, Association for Computational Linguistics, Oct. 2010.

[57] L. Richardson, "Beautiful soup documentation," *April*, 2007.

[58] M. R. Hossain and M. M. Hoque, "Towards Bengali word embedding: Corpus creation, intrinsic and extrinsic evaluations," in *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, (Indian Institute of Technology Patna, Patna, India), pp. 453–459, NLP Association of India (NLPAI), Dec. 2020.

[59] E. Loper and S. Bird, "NLTK: the natural language toolkit," *CoRR*, vol. cs.CL/0205028, 2002.

[60] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," vol. 16, p. 321–357, jun 2002.

[61] D. J. C. MacKay, "Hyperparameters: optimize, or integrate out?," in *Fundamental Theories of Physics: Their Clarification, Development and Application* (H. G.R., ed.), vol. 62, (Dordrecht: Springer), pp. 43–60, Springer, Dordrecht, 1996.

[62] K. Bostrom and G. Durrett, "Byte pair encoding is suboptimal for language model pretraining," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, (Online), pp. 4617–4624, Association for Computational Linguistics, Nov. 2020.

[63] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, (Valencia, Spain), pp. 1107–1116, Association for Computational Linguistics, Apr. 2017.

[64] X. Tao, Y. Li, R. Y. K. Lau, and H. Wang, "Unsupervised multi-label text classification using a world knowledge ontology," in *Advances in Knowledge Discovery and Data Mining*, (Berlin, Heidelberg), pp. 480–492, Springer Berlin Heidelberg, 2012.

[65] X. Zhang, J. Zhang, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proceedings of the 28th International Conference on Neural Information Processing Systems*, vol. 1, (Cambridge, MA, USA), pp. 649–657, MIT Press, 2015.

[66] J. Y. Lee and F. Dernoncourt, "Sequential short-text classification with recurrent and convolutional neural networks," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (San Diego, California), pp. 515–520, Association for Computational Linguistics, June 2016.

[67] E. Shriberg, R. Dhillon, S. Bhagat, J. Ang, and H. Carvey, "The ICSI meeting recorder dialog act (MRDA) corpus," in *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue at HLT-NAACL 2004*, (Cambridge, Massachusetts, USA), pp. 97–100, Association for Computational Linguistics, 2004.

[68] K. Xu, Y. Feng, S. Huang, and D. Zhao, "Semantic relation classification via convolutional neural networks with simple negative sampling," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, (Lisbon, Portugal), pp. 536–540, Association for Computational Linguistics, 2015.

[69] S. E. Saad and J. Yang, "Twitter sentiment analysis based on ordinal regression," *IEEE Access*, vol. 7, pp. 163677–163685, 2019.

[70] K. Kowsari, D. E. Brown, M. Heidarysafa, K. J. Meimandi, M. S. Gerber, and L. E. Barnes, "Hdltex: Hierarchical deep learning for text classification," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 364–371, 2017.

[71] J.-P. Mei, Y. Wang, L. Chen, and C. Miao, "Large scale document categorization with fuzzy clustering," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 5, pp. 1239–1251, 2017.

[72] Y. Xiao, B. Liu, J. Yin, and Z. Hao, "A multiple-instance stream learning framework for adaptive document categorization," *Know.-Based Syst.*, vol. 120, no. C, pp. 198–210, 2017.

[73] G. Nikolentzos, P. Meladianos, F. Rousseau, Y. Stavrakas, and M. Vazirgiannis, "Multivariate gaussian document representation from word embeddings for text categorization," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, (Valencia, Spain), pp. 450–455, Association for Computational Linguistics, Apr. 2017.

[74] M. A. Hearst, "Support vector machines," *IEEE Intelligent Systems*, vol. 13, no. 4, pp. 18–28, 1998.

[75] X. Li and D. Roth, "Learning question classifiers," in *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

[76] J. He, L. Wang, L. Liu, J. Feng, and H. Wu, "Long document classification from local word glimpses via recurrent attention learning," *IEEE Access*, vol. 7, pp. 40707–40718, 2019.

[77] S. Bahassine, A. Madani, and Mohamed, "Arabic text classification using new stemmer for feature selection and decision trees," *Journal of Engineering Science and Technology*, vol. 12, pp. 1475–1487, 2017.

[78] F. J. Tweedie, S. Singh, and D. I. Holmes, "Neural network applications in stylometry: The federalist papers," *Computers & Humanities*, vol. 30, pp. 1–10, January 1996.

[79] K. Luyckx, W. Daelemans, and E. Vanhoutte, "Stylogenetics: Clustering-based stylistic analysis of literary corpora," in *Proc. LREC'06*, (Genoa, Italy), pp. 30–35, 2006.

[80] S. Ruder, P. Ghaffari, and J. G. Breslin, "Character-level and multi-channel convolutional neural networks for large-scale authorship attribution," *CoRR*, vol. abs/1609.06686, September 2016.

[81] A. Rocha, W. J. Scheirer, C. W. Forstall, T. Cavalcante, A. Theophilo, B. Shen, A. R. B. Carvalho, and E. Stamatatos, "Authorship attribution for social media forensics," *IEEE Tran. on Information Forensics and Security*, vol. 12, pp. 5–33, January 2017.

[82] M. Koppel, J. Schler, and S. Argamon, "Authorship attribution in the wild," *Language Resources and Evaluation*, vol. 45, pp. 83–94, January 2010.

[83] E. Villar-Rodriguez, J. D. Ser, M. N. Bilbao, and S. Salcedo-Sanz, "A feature selection method for author identification in interactive communications based on supervised learning and language typicality," *Engineering Applications of Artificial Intelligence*, vol. 56, pp. 175–184, 2016.

[84] A. Hussein, N. Ghneim, and A. Joukhadar, "DamascusTeam at NLP4IF2021: Fighting the Arabic COVID-19 infodemic on Twitter using AraBERT," in *Proceedings of the Fourth Workshop on NLP for Internet Freedom: Censorship, Disinformation, and Propaganda*, (Online), pp. 93–98, Association for Computational Linguistics, June 2021.

[85] J. Mattern, Y. Qiao, E. Kerz, D. Wiechmann, and M. Strohmaier, "FANG-COVID: A new large-scale benchmark dataset for fake news detection in German," in *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*, (Dominican Republic), pp. 78–91, Association for Computational Linguistics, Nov. 2021.

[86] M. H. Saghayan, S. F. Ebrahimi, and M. Bahrani, "Exploring the impact of machine translation on fake news detection: A case study on persian tweets about covid-19," in *2021 29th Iranian Conference on Electrical Engineering (ICEE)*, pp. 540–544, 2021.

[87] R. Harakawa and M. Iwahashi, "Ranking of importance measures of tweet communities: Application to keyword extraction from covid-19 tweets in japan," *IEEE Transactions on Computational Social Systems*, vol. 8, no. 4, pp. 1030–1041, 2021.

[88] W. Anwar, I. S. Bajwa, M. A. Choudhary, and S. Ramzan, "An empirical study on forensic analysis of urdu text using lda-based authorship attribution," *IEEE Access*, vol. 7, pp. 3224–3234, December 2018.

[89] K. M. Jambi, I. H. Khan, M. A. Siddiqui, and S. O. Alhaj, "Towards authorship attribution in arabic short-microblog text," *IEEE Access*, vol. 9, pp. 128506–128520, 2021.

[90] M. Al-Sarem, F. Saeed, A. Alsaeedi, W. Boulila, and T. Al-Hadhrami, "Ensemble methods for instance-based arabic language authorship attribution," *IEEE Access*, vol. 8, pp. 17331–17345, January 2020.

[91] A. Taha, H. M. Khalil, and T. El-shishtawy, "A two level learning model for authorship authentication," *PLOS ONE*, vol. 16, pp. 1–16, 08 2021.

[92] M. A. Rahman and E. K. Dey, "Datasets for aspect-based sentiment analysis in bangla and its baseline evaluation," *Data*, vol. 03, no. 02, 2018.

[93] F. Kabir, S. Siddique, M. R. A. Kotwal, and M. N. Huda, "Bangla text document categorization using stochastic gradient descent (sgd) classifier," pp. 1–4, 2015.

[94] M. R. H. M. M. Hoque, "Automatic bengali document categorization based on word embedding and statistical learning approaches," pp. 1–6, 2018.

[95] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016.

[96] A. Ahmad and M. R. Amin, "Bengali word embeddings and it's application in solving document classification problem," pp. 425–430, 2016.

[97] M. R. Hossain and M. M. Hoque, "Automatic bengali document categorization based on deep convolution nets," vol. 882, pp. 513–525, 2019.

[98] A. Dhar, H. Mukherjee, S. M. Obaidullah, K. C. Santosh, N. S. Dash, and K. Roy, "Web text categorization: A lstm-rnn approach," vol. 1034, pp. 281–290, 2020.

[99] A. Khatun, A. Rahman, M. S. Islam, and Marium-E-Jannat, "Authorship attribution in bangla literature using character-level cnn," in *Proc. ICCIT*, (Dhaka, Bangladesh), pp. 1–5, 2019.

[100] A. K. Dipongkor, M. S. Islam, H. Kayesh, M. S. Hossain, A. Anwar, K. A. Rahman, and I. Razzak, "Daab: Deep authorship attribution in bengali," in *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9, 2021.

[101] M. R. Hossain, M. M. Hoque, M. A. A. Dewan, N. Siddique, N. Islam, and I. H. Sarker, "Authorship classification in a resource constraint language using convolutional neural networks," *IEEE Access*, vol. 9, pp. 100319–100338, 2021.

[102] M. M. Rahman, M. A. Pramanik, R. Sadik, M. Roy, and P. Chakraborty, "Bangla documents classification using transformer based deep learning models," in *2020 2nd International Conference on Sustainable Technologies for Industry 4.0 (STI)*, pp. 1–5, 2020.

[103] R. Das and T. D. Singh, "A multi-stage multimodal framework for sentiment analysis of assamese in low resource setting," *Expert Systems with Applications*, vol. 204, p. 117575, 2022.

[104] M. T. Alam and M. M. Islam, "Bard: Bangla article classification using a new comprehensive dataset," in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pp. 1–5, 2018.

[105] A. Kunchukuttan, D. Kakwani, S. Golla, G. N.C., A. Bhattacharyya, M. M. Khapra, and P. Kumar, "Ai4bharat-indicnlp corpus: Monolingual corpora and word embeddings for indic languages," *arXiv preprint arXiv:2005.00085*, 2020.

[106] github.com, "Stylogenetics," [Online]. Available: https://github.com/olee12/Stylogenetics.[Accessed: 7 Aug. 2018].

[107] taslimanasrin.com, "Taslima nasreen," [Online]. Available: https://www.taslimanasrin.com/category/bangla-blogs/. [Accessed: 12 Jan. 2020].

[108] banglapdfbooksblog.wordpress.com, "Bangla pdf books blog," [Online]. Available: https://banglapdfbooksblog.wordpress.com/. [Accessed: 14 Jan. 2020].

[109] www.goodreads.com, "Jibanananda das," [Online]. Available: https://www.goodreads.com/author/quotes/638250.Jibanananda_Das. [Accessed: 16 Jan. 2020].

[110] D. Kouzis-Loukas, *Learning Scrapy*. Packt Publishing Ltd, 2016.

[111] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37—46, 1960.

[112] O. Einea, A. Elnagar, and R. A. Debsi, "Sanad: Single-label arabic news articles dataset for automatic text categorization," *Data in Brief*, vol. 25, p. 104076, 2019.

[113] nltr.org, "Society for natural language technology research (snltr)," [Online]. Available: https://www.nltr.org/. [Accessed: 12 July 2018].

[114] ebanglalibrary.com, "Bangla book," [Online]. Available: https://www.ebanglalibrary.com/.[Accessed: 20 Aug. 2018].

[115] S. K. Dasari, A. Cheddad, J. Palmquist, and L. Lundberg, "Clustering-based adaptive data augmentation for class-imbalance in machine learning (cada): additive manufacturing use case," May 2022.

[116] S. Sen, A. Ekbal, and P. Bhattacharyya, "Parallel corpus filtering based on fuzzy string matching," in *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, (Florence, Italy), pp. 289–293, Association for Computational Linguistics, Aug. 2019.

[117] N. Ljubesic, T. Erjavec, and D. Fiser, "Datasets of Slovene and Croatian moderated news comments," in *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, (Brussels, Belgium), pp. 124–131, Association for Computational Linguistics, Oct. 2018.

[118] G. Palmer, V. Dâ€™Orazio, M. Kenwick, and M. Lane, "The mid4 dataset, 2002â€“2010: Procedures, coding rules and description," *Conflict Management and Peace Science*, vol. 32, no. 2, pp. 222–242, 2015.

[119] A. Weichselbraun, M. Göbel, and F. Odoni, "inscriptis," 2016.

[120] K. Luo, X. Luo, and K. Zhu, "Inferring binary relation schemas for open information extraction," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, (Lisbon, Portugal), pp. 555–560, Association for Computational Linguistics, Sept. 2015.

[121] D. M. Bikel and V. Castelli, "Event matching using the transitive closure of dependency relations," in *Proceedings of ACL-08: HLT, Short Papers*, (Columbus, Ohio), pp. 145–148, Association for Computational Linguistics, June 2008.

[122] K. Draszawka and J. Szymanski, "Thresholding strategies for large scale multilabel text classifier," in *2013 6th International Conference on Human System Interactions (HSI)*, pp. 350–355, 2013.

[123] T. Kieu, B. Vo, T. Le, Z.-H. Deng, and B. Le, "Mining top-k co-occurrence items with sequential pattern," *Expert Systems with Applications*, vol. 85, pp. 123–133, 2017.

[124] R. Lopez, T. Pardo, L. Avanco, P. Filho, A. Bokan, P. Cardoso, M. Dias, F. Nobrega, M. Cabezudo, J. Souza, A. Zacarias, E. Seno, and A. D. Felippo, "A qualitative analysis of a corpus of opinion summaries based on aspects," in *Proceedings of The 9th Linguistic Annotation Workshop*, (Denver, Colorado, USA), pp. 62–71, June 2015.

[125] T. Eckart, U. Quasthoff, and D. Goldhahn, "The influence of corpus quality on statistical measurements on language resources," in *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, (Istanbul, Turkey), pp. 2318–2321, European Language Resources Association (ELRA), May 2012.

[126] S. Terragni, E. Fersini, and E. Messina, "Word embedding-based topic similarity measures," in *Natural Language Processing and Information Systems*, (Cham), pp. 33–45, Springer International Publishing, 2021.

[127] G. Mikros, "Quantitative parameters in corpus design: Estimating the optimum text size in Modern Greek language," in *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, (Las Palmas, Canary Islands - Spain), pp. 834–838, European Language Resources Association (ELRA), May 2002.

[128] P. Shanta, L. Shibamouli, and B. Arindam, "Authorship attribution in bengali language," in *Proc. ICON*, (Trivandrum, India), pp. 100–105, 2015.

[129] H. A. Chowdhury, M. A. H. Imon, and M. S. Islam, "A comparative analysis of word embedding representations in authorship attribution of bengali literature," in *2018 21st International Conference of Computer and Information Technology (ICCIT)*, pp. 1–6, 2018.

[130] R. Rehearsal and P. Sojka, "Software framework for topic modelling with large corpora," in *Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks*, (Valletta, Malta), pp. 46–50, University of Malta, 2010.

[131] P. Madhyastha and R. Jain, "On model stability as a function of random seed," in *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, (Hong Kong, China), pp. 929–939, Association for Computational Linguistics, Nov. 2019.

[132] R. A. Potamias, G. Siolas, and A. G. Stafylopatis, "A transformer-based approach to irony and sarcasm detection," *Neural Computing and Applications*, vol. 32, pp. 17309–17320, Dec 2020.

[133] T. Gong, T. Fan, J. Guo, and Z. Cai, "Gpu-based parallel optimization of immune convolutional neural network and embedded system," *Engineering Applications of Artificial Intelligence*, vol. 62, pp. 384–395, 2017.

[134] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi, *et al.*, "Keras tuner." https://github.com/keras-team/keras-tuner, 2019.

[135] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Res.*, vol. 15, pp. 1929–1958, January 2014.

[136] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, October 1986.

[137] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, June 2017.

[138] D. E. Rumelhart and G. E. H. andRonald J Williams, *Learning Representations by Back-Propagating Errors*, pp. 696–699. Cambridge, MA, USA: MIT Press, 1988.

[139] A. F. M. Agarap, "Deep learning using rectified linear units (relu)," *CoRR*, vol. abs/1803.08375, 2018.

[140] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020.

[141] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," NIPS'17, (Red Hook, NY, USA), p. 6000–6010, Curran Associates Inc., 2017.

[142] X. Liu and C. Wang, "An empirical study on hyperparameter optimization for fine-tuning pre-trained language models," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, (Online), pp. 2286–2300, Association for Computational Linguistics, Aug. 2021.

[143] Z. Vitalii, S. Aleksandar, S. April, and H. Nils, "Correlation coefficients and semantic textual similarity," in *Proc. NACAL*, (Minneapolis, Minnesota), pp. 951–962, 2019.

[144] T. Mikolov, W. tau Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (Atlanta, Georgia), pp. 746–751, Association for Computational Linguistics, June 2013.

[145] O. Levy and Y. Goldberg, "Linguistic regularities in sparse and explicit word representations," in *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, (Ann Arbor, Michigan), pp. 171–180, Association for Computational Linguistics, June 2014.

[146] Y. Dai, L. Shou, M. Gong, X. Xia, Z. Kang, Z. Xu, and D. Jiang, "Graph fusion network for text classification," *Knowledge-Based Systems*, vol. 236, p. 107659, 2022.

[147] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning word vectors for 157 languages," in *Proc. LREC'2018*, (Miyazaki, Japan), pp. 3483–3487, 2018.

[148] J. Williams, R. Comanescu, O. Radu, and L. Tian, "DNN multimodal fusion techniques for predicting video sentiment," in *Proceedings of Grand Challenge and Workshop on Human Multimodal Language (Challenge-HML)*, (Melbourne, Australia), pp. 64–72, Association for Computational Linguistics, July 2018.

[149] M. R. Hossain and M. M. Hoque, "Automatic bengali document categorization based on word embedding and statistical learning approaches," in *Proc. IC4ME2*, (Rajshahi, Bangladesh), pp. 1–6, 2018.

[150] S. Lahiri, S. Phani, and A. Biswas, "A supervised learning approach for authorship attribution of bengali literary texts," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 16, August 2017.

[151] Q. Sun, J. Niu, Z. Yao, and H. Yan, "Exploring ewom in online customer reviews: Sentiment analysis at a fine-grained level," *Engineering Applications of Artificial Intelligence*, vol. 81, pp. 68–78, 2019.

[152] Y. Kim, "Convolutional neural networks for sentence classification," pp. 1746–1751, Oct. 2014.

[153] C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," vol. 2, no. 3, 2011.

[154] M. Alhawarat and A. O. Aseeri, "A superior arabic text categorization deep model (satcdm)," *IEEE Access*, vol. 8, pp. 24653–24661, 2020.

[155] X. Zhang and C.-A. Liu, "Model averaging prediction by k-fold cross-validation," *Journal of Econometrics*, 2022.

[156] M. R. Hossain and M. M. Hoque, "Covtexminer: Covid text mining using cnn with domain-specific glove embedding," in *Intelligent Computing & Optimization*, (Cham), pp. 65–74, Springer International Publishing, 2023.

[157] M. Kejriwal, K. Shen, C.-C. Ni, and N. Torzec, "Transfer-based taxonomy induction over concept labels," *Engineering Applications of Artificial Intelligence*, vol. 108, p. 104548, 2022.

[158] J. Han and H. Wang, "Transformer based network for open information extraction," *Engineering Applications of Artificial Intelligence*, vol. 102, p. 104262, 2021.

[159] M. T. Alam and M. M. Islam, "Bard: Bangla article classification using a new comprehensive dataset," in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pp. 1–5, 2018.

[160] M. M. Rahman, R. Sadik, and A. A. Biswas, "Bangla document classification using character level deep learning," in *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pp. 1–6, 2020.

[161] K. I. Islam, S. Kar, M. S. Islam, and M. R. Amin, "SentNoB: A dataset for analysing sentiment on noisy Bangla texts," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, (Punta Cana, Dominican Republic), pp. 3265–3271, Association for Computational Linguistics, Nov. 2021.

[162] M. R. Hossain, M. M. Hoque, and I. H. Sarker, "Text classification using convolution neural networks with fasttext embedding," in *Proc. HIS*, (Cham), pp. 103–113, Springer International Publishing, 2021.

[163] N. R. Bhowmik, M. Arifuzzaman, and M. R. H. Mondal, "Sentiment analysis on bangla text using extended lexicon dictionary and deep learning algorithms," *Array*, vol. 13, p. 100123, 2022.

[164] M. M. Rahman, M. A. Pramanik, R. Sadik, M. Roy, and P. Chakraborty, "Bangla documents classification using transformer based deep learning models," in *2020 2nd International Conference on Sustainable Technologies for Industry 4.0 (STI)*, pp. 1–5, 2020.

[165] S. Kula, R. Kozik, and M. Choras, "Implementation of the bert-derived architectures to tackle disinformation challenges," *Neural Computing and Applications*, Jul 2021.

[166] D. Kakwani, A. Kunchukuttan, S. Golla, G. N.C., A. Bhattacharyya, M. M. Khapra, and P. Kumar, "IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, (Online), pp. 4948–4961, Association for Computational Linguistics, Nov. 2020.

[167] B. L. WELCH, "THE GENERALIZATION OF 'STUDENT'S' PROBLEM WHEN SEVERAL DIFFERENT POPULATION VARLANCES ARE INVOLVED," *Biometrika*, vol. 34, pp. 28–35, 01 1947.

[168] T. Alam, A. Khan, and F. Alam, "Bangla text classification using transformers," *CoRR*, vol. abs/2011.04446, 2020.

[169] M. Rafi-Ur-Rashid, M. Mahbub, and M. A. Adnan, "Breaking the curse of class imbalance: Bangla text classification," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 21, apr 2022.

[170] M. La Quatra and L. Cagliero, "Transformer-based highlights extraction from scientific papers," *Knowledge-Based Systems*, vol. 252, p. 109382, 2022.

[171] M. Firdaus, A. Ekbal, and E. Cambria, "Multitask learning for multilingual intent detection and slot filling in dialogue systems," *Information Fusion*, vol. 91, pp. 299–315, 2023.

[172] M. Arbane, R. Benlamri, Y. Brik, and A. D. Alahmar, "Social media-based covid-19 sentiment classification model using bi-lstm," *Expert Systems with Applications*, vol. 212, p. 118710, 2023.

[173] Stehman and S. V., "Selecting and interpreting measures of thematic classification accuracy," *Remote Sensing of Environment*, vol. 62, pp. 77–89, Oct. 1997.

[174] S. M. Singh and T. D. Singh, "An empirical study of low-resource neural machine translation of manipuri in multilingual settings," *Neural Computing and Applications*, May 2022.

[175] P. B. Sphaier and A. Paes, "User intent classification in noisy texts: an investigation on neural language models," *Neural Computing and Applications*, May 2022.

Appendix

Table 1 displays four Bengali author texts extracted from four distinct corpora, along with the corresponding outputs generated by the proposed AFuNeT system.

Table 1: Sample inputs and output based on proposed AFuNeT model

Corpus	Input	Output	URL
LD	তাহার সেই বিরলশুভ্রকেশধারী সরলসুন্দরমুখচ্ছবি শান্তস্নেহহাস্যময় ধীপ্রদীপ্ত ঈশান দেখো গালমন্দ দিয়ো না বলছি তীরে লয়ে তাঁরে সিক্ত অঙ্গ পাইতেছে না তাহাকে কেন ভালোবাসে...	Actual: Bankim Chandra Chattopadhyay, Predicted: Bankim Chandra Chattopadhyay	https://tinyurl.com/3pfdxhbm
BAAD6	বছর কয়েক হল সাদিয়া আর আশরাফ সুন্দরবনের বইখালীতে জেলে শিশুদের জন্য একটি ...	Actual: Author-03, Predicted:Author-03	https://tinyurl.com/2p8fnsur
BACC-18	কুহেলিকা জাহাঙ্গীরের প্রিয়বন্ধু অনিমেষ বলিয়া উঠিল প্রমতদা জাহাঙ্গীরকে আমাদের দলে নেওয়ায় অন্তত আমার কোনো ...	Actual: Kazi Nazrul Islam, Predicted: Kazi Nazrul Islam	https://tinyurl.com/yc3t688r
BAAD16	তোমার– লাল চুলের মেয়েটি অপ্রকৃতস্থের মতো হাসতে থাকে। কিছুতেই হাসি থামাতে পারে না। সাতজনের ছোট দলটি নিঃশব্দে হেঁটে যেতে থাকে ...	Actual: MZI, Predicted: MZI	https://tinyurl.com/kh8yt3k7

Table 2 showcases four lengthy Bengali text classification samples extracted from diverse text classification corpora. The outputs are generated using AFuNeT.

Table 2: Sample inputs and output based on proposed AFuNeT model

Corpus	Input	Output	URL
BARD	চট্টগ্রাম ও ঢাকা থেকে ব্যাংকক এবং চট্টগ্রাম থেকে কলকাতা রুটে দুটি ফ্লাইট চালু করতে যাচ্ছে ইউএস ...	Actual: Economy, Predicted:Economy	https://tinyurl.com/2jf6ytx3
OSBC	আতাইকুলা উপজেলার শালাইপুর এলাকায় মঙ্গলবার ভোররাতে পূর্ব বাংলার কমিউনিষ্ট পার্টির (এম-এল) লাল পতাকা গ্র"পের পাবনা জেলা কমান্ডার শিমুল প্রামাণিক (৩৭) নিহত হয়েছেন ...	Actual: Accident, Predicted:Politics	https://tinyurl.com/yb75n7hs
BTCC11	রাজশাহীর উপজেলা স্বাস্থ্য কমপ্লেক্সে পুরুষ নার্সের (ব্রাদার) বিরুদ্ধে একজন রোগীর স্ত্রীকে যৌন হয়রানি করার অভিযোগ উঠেছে ...	Actual: Crime, Predicted:Crime	https://tinyurl.com/ye289hha
BCovC	করোনাভাইরাস: মক্কা-মদিনায় আবার বিধিনিষেধ আরোপ করেছে সৌদি সরকার...	Actual: COVID-19, Predicted:COVID-19	https://tinyurl.com/ycyhf36k

Table 3 displays three text samples sourced from distinct corpora, each containing a minimal number of words. The proposed system, unfortunately, fails to accurately predict the corresponding category names for these samples.

Table 3: Sample inputs and output based on proposed AFuNeT model

Corpus	Input	Output	URL
BTCC11	বিদ্যুৎস্পৃষ্ট হয়ে বিদ্যুতের মৃত্যু	Actual: Accident, Predicted:Crime	https://tinyurl.com/ykj9txhc
BARD	তরুণ বয়সে হেভিওয়েট বক্সার ছিলেন নেলসন ম্যান্ডেলা।	Actual: Sports, Predicted: International	https://tinyurl.com/5n74bvta
BCovC	১২শ কোটি টাকার প্রণোদনা আবেদন	Actual: COVID-19, Predicted:Non-COVID-19	https://tinyurl.com/3x48zkc7