Studies on Bengali Text Classification Using Machine Learning and Deep Learning

Thesis submitted in partial fulfillment of requirements
For the degree of
Master of Technology in Computer Technology
of
Computer Science and Engineering Department
of
Jadavpur University

by

Amartya Roy
Regn. No.- 154171 of 2020 - 2021
Exam Roll No. - M6TCT23001

under the supervision of

Dr. Chinatan Kumar Mandal
Department of Computer Science and Engineering
Jadavpur University
Kolkata, West Bengal, India
2023

Certificate from the Supervisor

This is to certify that the work embodied in this thesis entitled **"Studies on Bengali Text Classification Using Machine Learning and Deep Learning"** has been satisfactorily completed by **Amartya Roy** (Registration Number 154171 of 2020−2021; Class Roll No. 002010504005; Examination Roll No. M6TCT23001). It is a bona-fide piece of work carried out under my supervision and guidance at Jadavpur University, Kolkata for partial fulfilment of the requirements for of the degree of **Master of Technology in Computer Technology** in the **Department of Computer Science and Engineering, Jadavpur University**, during the period of September 2022 to June 2023

Chintan Kumar Mandal,
Assistant Professor,
Department of Computer Science and Engineering,
Jadavpur University.
(Supervisor)

Forwarded By:

Prof. Nandini Mukherjee,
Head,
Department of Computer Science and Engineering,
Jadavpur University.

Prof. Ardhendu Ghoshal,
DEAN,
Faculty of Engineering & Technology,
Jadavpur University.

Department of Computer Science and Engineering
Faculty of Engineering And Technology
Jadavpur University, Kolkata - 700 032

Certificate of Approval

This is to certify that the thesis entitled **"Studies on Bengali Text Classification Using Machine Learning and Deep Learning"** has been is a bona-fide record of work carried out by **Amartya Roy** (Registration Number 154171 of 2020 − 2021; Class Roll No. 002010504005; Examination Roll No. M6TCT23001). in partial fulfilment of the requirements for the award of the degree of **Master of Technology in Computer Technology** in the **Department of Computer Science and Engineering, Jadavpur University**, during the period of September 2022 to June 2023. It is understood that by this approval, the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the thesis only for the purpose of which it has been submitted.

Examiners:

(Signature of The Examiner)

(Signature of The Supervisor)

Department of Computer Science and Engineering
Faculty of Engineering And Technology
Jadavpur University, Kolkata - 700 032

Declaration of Originality and Compliance of Academic Ethics

I hereby declare that the thesis entitled **"Studies on Bengali Text Classification Using Machine Learning and Deep Learning"** contains literature survey and original research work by the undersigned candidate, as a part of his degree of **Master of Technology in Computer Technology** in the **Department of Computer Science and Engineering, Jadavpur University**. All information have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: Amartya Roy

Examination Roll No.: M6TCT23001

Registration No.: 154171 of 2020 – 2021

Thesis Title: Studies on Bengali Text Classification Using Machine Learning and Deep Learning

Signature of the Candidate:

ACKNOWLEDGEMENT

I am pleased to express my gratitude and regards towards my Project Guide **Dr. Chintan Kumar Mandal** Assistant Professor, and co-guide **Prof. Kamal Sarkar**, Department of Computer Science and Engineering, Jadavpur University, without whose valuable guidance, inspiration and attention towards me, pursuing my project would have been impossible.

Last but not the least, I express my regards towards **parents**, some of my best friends for bearing with me and for being a source of constant motivation during the entire term of the work.

Amartya Roy
MTCT Final Year
Exam Roll No. - M6TCT23001
Regn. No. - 154171 of 2020 – 2021
Department of Computer Science and Engineering,
Jadavpur University.

Contents

List of Figures

List of Tables

Abstract

This study focuses on Bengali text classification using machine learning and deep learning techniques. The research explores the efficacy of multilingual pre-trained models for Bengali text classification tasks and proposes algorithms to address challenges related to base classifier selection and input length constraints in existing models. The findings contribute to the advancement of Bengali text classification techniques by identifying effective pre-trained models, optimizing ensemble model selection, and bypassing the input length constraint of BERT models. The outcomes of this research have practical implications for improving the accuracy and efficiency of Bengali text classification in various applications.

Chapter 1

Introduction

As the world becomes increasingly dependent on technology, humans have pushed the limits of what they can think by fusing human and artificial brains. The outcome of this technique, which has achieved enormous appeal worldwide, is artificial intelligence (AI). Classification of text research is significantly impacted by the AI discipline of natural language processing (NLP). It is one of the well-known methods for locating, examining, comprehending, and collecting data from text-based sources. Each author publishes the information in their own native tongue, such as Spanish, Bangla, Chinese, or English, among the many other human languages that are available. However, it is the job of the computer to decipher and understand these writings, and NLP gives computers the ability to do so more intelligently. NLP has been much more well-liked in recent years. This method is used in many different applications, such as speech tagging, information extraction, and text classification. Text categorization uses NLP to group documents into a predetermined set of categories. However, due to the high dimensionality of the feature vector, which comprises irrelevant and unrelated data, text classification is a difficult job. In order to eliminate extra features and reduce the feature vector's dimension, a variety of feature reduction techniques have been proposed. To improve classification outcomes, a machine learning model employs a meaningful and condensed feature vector.

Bengali, one of the most widely spoken languages in the world, has garnered

significant attention in the field of natural language processing (NLP) and text classification. The increasing availability of digital content in Bengali has necessitated the development of effective techniques for organizing, analyzing, and extracting insights from Bengali textual data. Machine learning and deep learning approaches have emerged as powerful tools for Bengali text classification, enabling the automatic categorization of Bengali documents into predefined classes or topics.

Previous text categorization research relied on conventional machine learning algorithms. In order to do this, features like the bag of words or n-grams had to be manually chosen. These features were then inputs for classification algorithms like Naive Bayes (NB), Support Vector Machine (SVM), Hidden Markov Models (HMM), Random Forests, etc. However, the emphasis has switched to the use of deep learning techniques as large scale datasets have been more accessible in recent years. Deep learning models no longer require feature engineering and can be applied to a variety of tasks because they can learn representation from the data itself.

More recently, pre-trained language models based on transformers have been effectively utilised to learn language representations using a significant amount of unlabeled input. BERT, RoBERTa, OpenAI GPT, and others are a few of these models. When optimised for various downstream tasks, such as text categorization [25], question answering [12], natural language inference [27], etc., these models have demonstrated to be incredibly successful. However, these models are typically trained on big multilingual datasets that may contain over a hundred languages or on large monolingual English corpora.

Motivated by these and the fact that, there has not been many works on Bengali text classification based on this approach, We have done a comprehensive study on Bengali text classification using both machine learning and deep learning techniques

The contributions of this work are summarized as follows:

Exploration of Multilingual Pre-trained Models: In this study, we investigate the efficacy of different multilingual pre-trained models for Bengali text classification tasks. Multilingual models, such as BERT (Bidirectional Encoder Representations from Transformers), have been trained on large-scale multilingual corpora and have

shown promising results in various languages. We evaluate the performance of these models on Bengali text classification and analyze their suitability for capturing the linguistic nuances and characteristics of the Bengali language.

Algorithm for Base Classifier Selection: For traditional ensemble models, selecting suitable base classifiers can be a challenging task, particularly when dealing with Bengali text classification. In this work, we propose an algorithm for base classifier selection using the backward elimination method. The algorithm iteratively eliminates underperforming classifiers from the ensemble, based on their contribution to the overall performance. This approach aims to improve the efficiency and effectiveness of the ensemble model by selecting the most relevant and competent base classifiers.

Algorithm for Bypassing Input Length Constraint of BERT Models: BERT models, although powerful in capturing contextual information, often come with input length constraints, posing challenges when dealing with longer Bengali text sequences. To address this constraint, we propose an algorithm that bypasses the input length limitation of BERT models. By intelligently dividing longer text sequences into smaller segments and leveraging the contextual embeddings from BERT, we ensure that the classification model can effectively handle lengthier Bengali texts.

Chapter 2

Literature Survey on Bengali Text Classification

Text classification is emphasised in numerous articles, publications, and studies. In this chapter, we have discussed several prior research works that are pertinent to our thesis. Despite being one of the most widely spoken and culturally diverse languages with around 265 million native speakers, there hasn't been as much research on Bengali text classifications as there has been for English. The lack of labelled data for machine learning model training is the primary cause.

Because of its efficiency and simplicity in employing the joint probabilities of words and categories given a document, the Naive Bayes (Nave Bayes) probabilistic classifier is frequently used in text classification applications and experiments. Naive Bayes was employed by Abu Nowshed Chy et al [6] for the classification of Bangla news. In their study, Dhar et al. [8] introduced an auto-text classifier that can classify a Bangla text document. To categorise the text document, they proposed the TF-IDF-ICF unique feature selection approach. In their study, Kabir et al. [16] introduced the SGD categorizer, which classifies Bangla documents. from BDNews24, an online Bengali newspaper. They examined materials written in Bangla and divided them into nine groups. They had proposed a system that is broken down into three steps: classifier construction using SGD, characteristics expulsion with TF and IDF, and final measurement of perfection using the F1 score. For the goal of classifying texts,

Dhar et al. [10] evaluated the efficacy of different classifiers based on two feature expulsion procedures. Eight different text categories were present in the over 8000 text records taken from the online news corpus of Bengali publications, web pages, and websites for this experiment. They used WEKA for testing and constructed a total of 5 pattern recognition classifiers, including MLP, RF, SVM, NBM, and K* for nondecreased and decreased feature vectors. Using MLP for a non-reduced list of capabilities and a mean accuracy of 0.987 across all classifiers allowed for the highest level of precision. In their research, Dhar et al. [9] focused on developing a method for nurturing a computerised framework to address the problem of classifying Bangla text records into their unique text categories. For this test, a total of 8000 pieces of Bangla text across eight categories were taken into consideration from online news archives, specialty pages, and a few Bangla online distributions. For the layout of Bangla text reports based on common component extraction techniques, such as "term association" and "term aggregation," there existed a crossover procedure (PART). The dataset was subjected to five overlap cross-approvals, which produced a precision for the PART characterisation technique of 96.83 percent. SVM, NB, and SGD were three notable controlled learning algorithms that Islam et al. [14] introduced in their work to classify Bengali text. In this study, unique feature selection techniques, such as Chi-square distribution and standardised TF-IDF with word analyzer, are used to assess how well classifiers performed while predicting reports against twelve categories. Whole papers were constructed from a Bengali report corpus for the test, and 31908 pieces of information were obtained from various Bangladeshi media. After analysing the outcomes of various techniques, it was discovered that the SVM classifier had the most notable F1 score, 92.56 percent, using standardised TFIDF as component determination. To categorise Bangla web documents, Mandal and Sen [20] employed four machine learning algorithms, including Support Vector Machine (SVM), Naive Bays, K-Nearest Neighbour (KNN), and Decision Tree. The dataset they used for their research was the BD corpus, which has 1000 documents and a total of 22,218 words. It is divided into five categories: business, sports, health, technology, and education. To extract important features from the documents, they used the tf-idf feature extraction technique. They employ assessment metrics like recall, accuracy, and F-measure to gauge how well the classi-

fiers are working. SVM fared the best in their experiment with an average accuracy of 89.14 percent, and KNN performed the worst with an average accuracy of 74.24 percent. Additionally, they displayed the training times for each classifier, with SVM having the quickest training times.

Chapter 3

Background on Ensemble Learning & Pretrained Language Model

3.1 Background On Ensemble Method

There exists a number of ML methods that have been applied to text classification. But it has been shown[cite] that ensemble methods are more effective than the individual classifiers in text classification. Generally, ensemble methods may be classified as dependent or independent methods based on how each baseline classifier interacts with the others. In the dependent method, the result of one classifier has an impact on the formation of the next classifier. Boosting algorithms (Mayr et al [21]) are the most well-known of examples of dependent methods. The independent method, on the other hand, builds each classifier separately from subsets of the dataset and then combines the results in some way. According to Chandra and Yao [5], Liu and Yao [19], classifiers should ideally be independent or negatively correlated to make an effective ensemble. Random forest (ensemble of decision Tress) (Breiman [3]) and Stacking (Haghighi and Omranpour [13]) are the wide examples, among many others, of independent methods. The general framework of any ensemble learning in the independent method is to use an aggregation function G to combine a set k of baseline classifiers, $c_1, c_2, \ldots, c_k$, towards predicting a single output. Given a dataset of size n and features of dimension $m, D = \left\{(x_i, y_i)\right\}, 1 \leq i \leq n, x_i \in R^m$,

the predication of the output based on this method is given by 3.1.

$$y_i = \phi(x_i) = G(c_1, c_2, \ldots, c_k) \tag{3.1}$$

where $y_i \in Z$ denotes classification. Building an ensemble model, given this general framework, entails deciding how to train baseline classifiers and a suitable process for aggregating the outputs of baseline classifiers. For their successful improvement on predictive accuracy and easily parallelized training, several independent ensemble methods have been proposed over the last few years (Sagi and Rokach [24]; Rokach [23]; Polikar [22]). Each ensemble method requires a proper fusion of several learners in order to generate the final prediction model. Generally, fusion techniques can be divided up further into averaging and meta-learning techniques.

3.2 Background On Pretrained Language Model

Pretrained language models have revolutionized the field of NLP by providing powerful representations of text, reducing the need for extensive hand-crafted features, and achieving state-of-the-art results on various benchmarks. They have been widely adopted and used in academia and industry for a wide range of NLP applications, including chatbots, language translation, content generation, sentiment analysis, and more. These models have significantly advanced the capabilities of natural language understanding and generation systems. Pretrained language models, such as Transformer, BERT [7](Bidirectional Encoder Representations from Transformers), GPT [28] (Generative Pretrained Transformer), and RoBERTa [18] (Robustly Optimized BERT Approach), have achieved remarkable success in a wide range of natural language processing (NLP) tasks. These models capture rich contextual information, enabling them to understand the meaning of words in the context of a sentence or document. In our experiments, we utilize transformer language models[Vaswani et al. [26] & Dong et al. [11]] that are publicly accessible through HuggingFace's transformer library. In this section, we will give a brief overview of the transformer. We would like to recall the transformer architecture as given in Provable Memorization Capacity Of Tansformers

A. General Notation - Denote the number of input-output pairs as N, the number of output classes as C, the token embedding dimension as d, and the sequence length as n. We use σ_R to represent the ReLU activation function. We let σ_S and σ_H be the softmax and hardmax operators, respectively. These operators take a matrix as an input, apply softmax/hardmax column-wise, and output a column stochastic matrix of the same size. For $m \in N$, we define $[m] = \{1, 2, \cdots, m\}$. We use $|\mathcal{S}|$ to denote the number of elements in a set $\mathcal{S}$. We denote a set or a function as an uppercase calligraphic letter, a matrix by an uppercase bold letter and a vector by a lowercase bold letter.

B. Transformer Architecture - We define a Transformer $\mathcal{N} : R^{d\times n} \to R^{1\times n}$ of depth L as a composition of L Transformer blocks with input and output embedding mappings:

$$\mathcal{N} = \mathcal{E}_{\text{out}} \circ \mathcal{F}_L \circ \cdots \circ \mathcal{F}_2 \circ \mathcal{F}_1 \circ \mathcal{E}_{\text{in}}$$

where each Transformer block $\mathcal{F}_l : R^{m\times n} \to R^{m\times n}$ is a sequence-to-sequence function consisting of two subblocks: a self-attention subblock and a tokenwise feedforward subblock. The input embedding block $\mathcal{E}_{\text{in}} : R^{d\times n} \to R^{m\times n}$ and the output embedding block $\mathcal{E}_{\text{out}} : R^{m\times n} \to R^{1\times n}$ are 1-layer tokenwise linear mappings.

The self-attention subblock represents the interaction among tokens. Formally, given an input $Z \in R^{m\times n}$, the self-attention subblock $\mathcal{F}_l^{(SA)}$ with h heads and head size k computes

$$\mathcal{F}_l^{(SA)}(\boldsymbol{Z}) = \boldsymbol{Z} + \sum_{i=1}^{h} \boldsymbol{W}_{l,i}^{(O)} \left(\boldsymbol{W}_{l,i}^{(V)} \boldsymbol{Z}\right) \sigma_S \left[\left(\boldsymbol{W}_{l,i}^{(K)} \boldsymbol{Z}\right)^T \left(\boldsymbol{W}_{l,i}^{(Q)} \boldsymbol{Z}\right)\right]$$

where $\boldsymbol{W}_{l,i}^{(O)} \in R^{m\times k}$ and $\boldsymbol{W}_{l,i}^{(V)}, \boldsymbol{W}_{l,i}^{(K)}, \boldsymbol{W}_{l,i}^{(Q)} \in R^{k\times m}$ are the weight matrices parametrizing the self-attention subblock. We include a skip connection in the self-attention subblock. The feedforward subblock processes each token independently in parallel by applying two feedforward layers. Given an input $\boldsymbol{H} \in R^{m\times n}$, the feedforward subblock $\mathcal{F}_l^{(FF)}$ with dimension q computes

$$\mathcal{F}_l^{(FF)}(\boldsymbol{H}) = \boldsymbol{H} + \boldsymbol{W}_l^{(2)} \sigma_R \left(\boldsymbol{W}_l^{(1)} \boldsymbol{H} + \boldsymbol{b}_l^{(1)} 1_n^T\right) + \boldsymbol{b}_l^{(2)} 1_n^T$$

where $\boldsymbol{W}_l^{(2)} \in R^{m\times q}, \boldsymbol{b}_l^{(2)} \in R^m, \boldsymbol{W}_l^{(1)} \in R^{q\times m}$ and $\boldsymbol{b}_l^{(1)} \in R^q$ parametrize the feedforward subblock. The feedforward subblock also includes a skip connection. Finally, the Transformer block composes two subblocks as

$$\mathcal{F}_l(\boldsymbol{Z}) = \mathcal{F}^{(FF)}\left(\mathcal{F}^{(SA)}(\boldsymbol{Z})\right)$$

Since each Transformer block consists of a fixed number of layers even in the most fine-grained sense, we use the number of blocks L as the depth of the network. We would like to recall the defination of the width of the network as $\max\{m, kh, q\}$ where m is the embedding dimension , h is the number of attention heads, k is the attention head size, and q is the feedforward dimension

Chapter 4

Methodology

In this chapter, we discuss the methodology employed for Bengali text classification. The goal of this study is to develop effective machine learning and deep learning models for classifying Bengali text documents. This section provides an overview of the proposed machine learning and deep learning-based text classification models.

4.1 Proposed Machine Learning Based Text Classification

In this section, we will present the machine learning-based models proposed for Bengali text classification. Machine learning algorithms have been widely used for text classification tasks and have shown promising results. We have explored different machine learning algorithms such as Support Vector Machines (SVM), Random Forest, and Logistic Regression, Decision Trees, KNN, Naive Bayes. These algorithms have been trained on a labelled dataset consisting of Bengali text documents, and their performance have been evaluated using appropriate evaluation metrics.

The proposed ML-based classification approach has several steps such as 1) Preproccessing 2) Document Representation 3) Classification

4.1.1 Preprocessing

Our pre-processing process is divided into two steps. They are Stopword Removal and Stemming.

Stopword Removal:- Stop words are low-content words that occur frequently in a language. For example(in Bengali) অবশ্য, অনেক, অনেকে,অনেকেই,একে,একটি,এখন etc . The reason for stop word removal is to help a classification algorithm to focus on the important words when it classifies documents. We have used the NLTK library for removing stop words.

Stemming :- Inflection leads to word mismatch problem. Since Bengali is a highly inflectional language, we have used a Bengali stemmer to convert a word to its root. Though stemming does not produce the real roots of words, it is useful in document classification because it alleviates the data sparseness problem. Table 4.1 shows sample results of word stemming

Table 4.1: Example of Stemming

Original Words	**After Stemming**
এটাই	এটা
সেটাই	সেটা
হয়তো	হয়
হেসেছিলেন	হাসা

4.1.2 Document Representation

In our work, we have used the TF-IDF-based vector space model(VSM) for document representation. Let the given documents be denoted as $D = d_1, d_2, \ldots, d_n$ and the term $w = w_1, w_2, \ldots, w_m$ occurring in $d_1, d_2, \ldots, d_n$ and the count of a word w in document D is indicated by $count_{w,d}$.

TF(Term Frequency) is calculated as,

$$TF(w, d) = log(count_{w,d} + 1)$$

IDF(Inverse Document Frequency) is computed using the formula:

$$IDF(w, D) = log(\frac{N}{|d \in D : w \in d|})$$

where

- N : corpus size
- $|d \in D : w \in d|$: how many documents contain the word w occurs.
- If the term is absent from the document, the IDF becomes undefined. To avoid this, we generally convert it to $1 + |d \in D : w \in d|$.

The TF-IDF value is obtained in terms of TF and IDF in the following way:-

$$TFIDF(w, d, D) = TF(w, d) * IDF(w, D)$$

We represent each document as a n-dimensional vector v_d whose each component corresponds to the TF-IDF value of a dictionary term. Here a dictionary contains n distinct words derived from the entire corpus of the documents.

4.1.3 Classification

In our proposed ensemble method, we combine outputs of several base classifiers. The base classifiers are combined using the non-trainable classifier combination strategy. The most common such classifier combination strategies that we have tried for our implementation are average of probability, weighted average of probability and stack ensemble. Details of classifier combination strategies will be discussed later in this section. Architecture of our ensemble model is shown in Figure 4.1

Our proposed model is basically a hybrid classification model, which is implemented by developing the base classification models first and then combining the predictions of the developed base classifiers. The base classifiers that we have incorporated in our proposed ensemble model are Support Vector Classifier, Decision Tree, Random Forest, KNN, Logistic Regression and Naive Bayes. In the subsequent

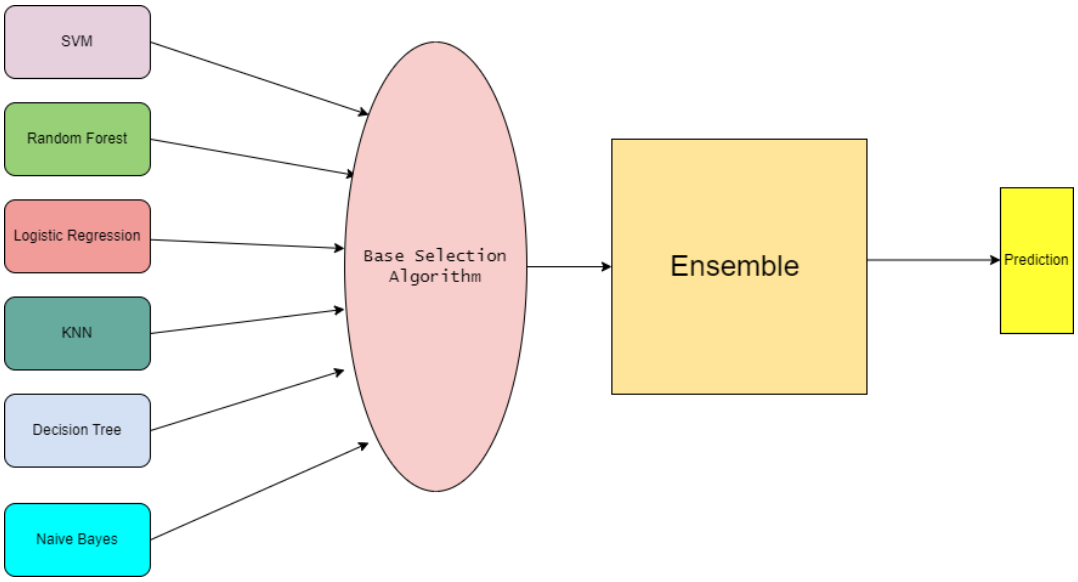


Figure 4.1: General Architecture Of Ensemble Model

sub-sections, we will discuss the details of the various base classifiers and also will discuss how the base models are chosen for our ensemble model.

Base (Machine Learning) Models

Here we have give a brief overview of all the base classifiers we have used for our experiment.

1. **Support Vector Machine** - Support Vector Machine (SVM) is a supervised learning model used for regression analysis and classification tasks. It aims to find a boundary, called a hyperplane, in a 2-dimensional space that separates data points belonging to different classes in a dataset. The goal is to find a hyperplane that maximizes the distinction between the data points of different classes in an n-dimensional space. The data points closest to this hyperplane are referred to as Support Vectors.

For binary classification, the hyperplane equation is given by:

$$w \cdot x - b = 0$$

Here, w represents the normal vector to the hyperplane, b is a threshold term, and x is the input instance. If the equation evaluates to a positive value, the instance belongs to one class, while if it evaluates to a negative value, the instance belongs to the other class.

Normally, SVM supports binary classification, but it can be extended to handle multiclass classification using two approaches: One-vs-One and One-vs-All.

In our thesis, we utilized the scikit-learn API, which employs the One-vs-One approach. This approach breaks down the multiclass problem into multiple binary classification problems, with each pair of classes having its own binary classifier. Therefore, for classifying data points from a dataset with n classes, the classifier employs $\frac{n \cdot (n-1)}{2}$ SVMs. In our case, we have two datasets with 6 and 38 classes, resulting in the usage of 15 and 703 SVMs, respectively.

2. **Random Forest** - The random forest algorithm is a popular supervised learning method that generates a forest consisting of multiple trees. It can be used for regression and classification problems, although it is commonly applied to solve classification problems. This approach involves creating a collection of decision trees where the best feature is selected from a random subset of features for each tree. Random forest is composed of these trees, which collectively classify new objects based on a given input vector.

In random forest, all the decision trees are utilized to classify data. Each tree "votes" for the class and the final classification is determined by the majority vote from all the trees. However, there are two factors that can contribute to potential errors in random forest. Firstly, if there is a correlation among trees within the forest, the error rate may increase. Secondly, each tree carries its own weight, meaning a tree with a lower error rate is considered more powerful as a classifier.

3. **Logistic Regression** - Logistic regression is one of the well-renowned supervised methods for text categorization. LR was frequently utilized in the realm of data retrieval. In past research, Logistic regression has been investigated in the machine learning area, specifically in English text categorization. In Logistic Regression the main focused word is "logistic" which refers to a logistic function that performs

classification operations in the algorithm. In addition, the Logistic function is the foundation of the logistic regression model. The following figure (3.1) [36] shows a logistic or sigmoid curve and the equation 4.1 is:

$$f(x) = \frac{M}{1 + e^{-k(x-x_0)}} \tag{4.1}$$

Here, e = Euler's number, $x_0 = x$ value of the sigmoid's midpoint, M = maximum value of curve, k = steepness or logistic rate of the curve and $f(x)$ = function output.

Logistic Regression is used to classify categorical dependent data by utilizing predictor variables. It converts probability scores from categorical dependent variables, hence creating a connection between a categorical variable which is dependent and a continuous variable which is independent. Binary logistic regression and multinomial logistic regression are two forms of logistic regression. The primary distinction between these two sorts is the type of labels used in them. Multinomial logistic regression is performed when the labels include several values. The model's of fundamental assumption is that the independent and dependent variables do not share a linear relationship [9]. When it comes to text classification the Logistic regression model identifies a vector including variables, assesses each input variable coefficients, and then assumes the text class using a word vector

4. **KNN** - K-Nearest Neighbors (KNN) is a non-parametric and instance-based classification algorithm. It operates based on the principle that similar instances tend to belong to the same class. In KNN, the "k" refers to the number of nearest neighbors used to make a prediction. Given a new input, KNN finds the k closest labeled instances in the training data based on a distance metric (e.g., Euclidean distance) and assigns the majority class label among those neighbors as the predicted label for the new instance. KNN is simple to implement, but its performance can be sensitive to the choice of k and the distance metric.

5. **Decision Tree**- A Decision Tree is a supervised machine learning algorithm used for classification and regression tasks. It creates a tree-like model where each internal node represents a feature or attribute, each branch represents a decision rule,

and each leaf node represents the predicted class or value. Decision Trees partition the feature space based on the values of input features and recursively split the data into smaller subsets until a stopping criterion is met (e.g., purity of classes or depth of the tree). The splitting process is guided by various impurity measures such as Gini Index or Information Gain. Decision Trees are interpretable and can handle both categorical and numerical features. However, they may suffer from overfitting and can be sensitive to small changes in the training data

6. **Naive Bayes** - Naive Bayes is a probabilistic classifier based on Bayes' theorem and assumes that the features are conditionally independent given the class label. It is known as "naive" because this assumption of independence is usually not true in real-world scenarios. Despite this simplification, Naive Bayes can perform well and is computationally efficient. It calculates the probability of each class for a given input by multiplying the prior probabilities of the classes with the conditional probabilities of each feature given the class. The class with the highest probability is assigned as the predicted class. Naive Bayes is particularly effective when there is a large number of features and relatively small training data. It is commonly used for text classification tasks, such as spam detection or sentiment analysis.

Classifier Combination Methods

In our case, we have initially considered a list of ML classifiers as the base learners. We have used **Base Classifier Selection Algorithm** for selecting the suitable base classifiers to improve the classifier accuracy. The **Base Classifier Selection Algorithm** is discussed later in this paper. The common classifier combination methods are 1) Average of probability, 2) Weighted average of probability and 3) Meta-learning ensemble. We have considered all these combination methods in our experiments. The classifier combination methods are briefly discussed below.

A. i) Averaging Ensemble - The Average of Probabilities ensemble method is a technique used to combine the predicted probabilities from multiple base classifiers to obtain a final prediction. Instead of combining the actual predictions, this method

focuses on aggregating the probabilities assigned to each class by the base classifiers. We have shown a general outline of this method in fig 4.2. In this ensemble method, multiple base classifiers are trained on the given training set. After training the base classifiers, they are used to predict the probabilities of each class for a given new data. The predicted probabilities represent the confidence of each classifier in assigning the instance to different classes.The ensemble method calculates the average of the predicted probabilities for each class across all the base classifiers. This is done by summing up the probabilities for each class and dividing by the total number of base classifiers. The class with the highest average probability is selected as the final prediction. For averaging ensemble we consider uniform weights.

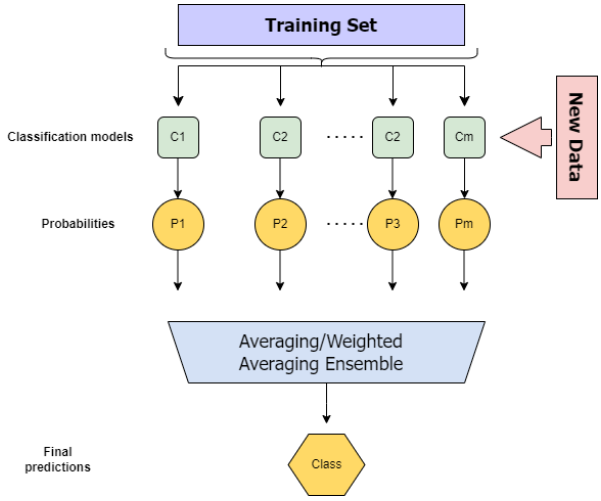


Figure 4.2: General Architecture Of Averaging Ensemble/ Weighted Averaging Ensemble For Text Classification

ii) Weighted Averaging Ensemble - The weighted average ensemble method is a technique used to combine the predictions of multiple base classifiers by assigning weights to each classifier based on their individual accuracies. The weights are calculated proportionally to the accuracies of the classifiers. Here's the formal definition of the weighted average ensemble method: Suppose we have three base classifiers c1, c2, c3. and accuracies of the each base classifiers are a1, a2, a3. Then the weights for each classifier will be $weight\_c1 = \frac{a1}{a1+a2+a3}$, $weight\_c2 = \frac{a2}{a1+a2+a3}$

and $weight\_c3 = \frac{a3}{a1+a2+a3}$.

The weighted average prediction will be

$$\begin{aligned} weighted\_prediction = {} & weight\_c1 \cdot prediction\_c1 \\ & + weight\_c2 \cdot prediction\_c2 \\ & + weight\_c3 \cdot prediction\_c3 \end{aligned}$$

The weighted average ensemble method assigns higher weights to classifiers with higher accuracies, indicating that they contribute more to the final prediction. By combining the predictions of multiple classifiers, the ensemble method aims to improve the overall accuracy and robustness of the model.

B. Meta-learning Ensemble- Meta-learning is a technique that involves learning from other classifiers in order to improve overall performance. Unlike traditional learners, meta-training classifiers undergo multiple learning steps instead of just one. We have shown a general architecture of this ensemble method in fig 4.3. The process begins by training baseline classifiers, which are then used to train a meta-classifier responsible for combining the predictions of the baseline classifiers. During the prediction phase, the baseline classifiers provide their individual classifications, and the meta-classifier performs the final classification.

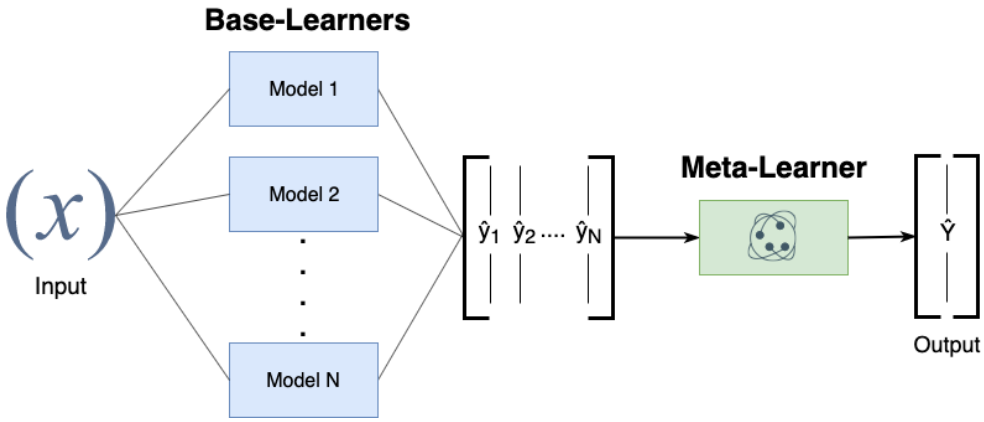


Figure 4.3: General Architecture Of Stacking For Text Classification

Stacking is a well-known meta-learning method that follows a two-stage classification structure, consisting of the baseline classifiers and the meta-classifier. The motivation behind stacking arises from the limitations of simple average ensembles, where each model's contribution is treated equally regardless of its performance. In contrast, stacking introduces a higher-level model that combines the predictions of the individual models. Each model in the ensemble is trained individually using the same training set, often referred to as the Level-0 training set. The predictions generated by these models are then used to construct a Level-1 training set.

Selection of Base (Machine Learning) Models

We have used base classifiers as Support Vector Classifier, Decision Tree, Random Forest, KNN, Logistic Regression, Naive Bayes. Among these, which models will be selected as base classifiers of the ensemble model of highest accuracy that gives birth to a question of careful selection, that's why to suitably select these base classifiers we have proposed **Base Classifier Selection Algorithm Using Backward Elimination Method**, and we have selected our best possible base classifiers using this algorithm 1. For our case, best base classifiers are SVC, Random Forest and Linear Regression. Hence we have used these base classifiers in our ensemble methods discussed in section 4.1.3 for getting final prediction.

Algorithm 1: Base Classifier Selection Algorithm Using Backward Elimination Method

Input : A set of n trained classifiers
Output: A subset of n classifiers

Order the classifiers based on their accuracy on the validation set(from highest to lowest accuracy) and prduce an ordered list of classifiers, CL
Create an ensemble with n classifiers
Calculate the accuracy of the ensemble, ACC
repeat
    Simulate removing $CL[n]$ from CL
    Create an ensemble of CL_{n-1} classifiers
    Calculate the accuracy of the ensemble, $ACC_{CL_{n-1}}$
    if $ACC_{CL_{n-1}} \geq ACC$ **then**
        $CL[n]$ is removed from CL
    else
        $CL[n]$ is not removed from CL
    end if
    $n = n - 1$
until $n = 0$;
return CL

4.2 Deep Learning Based Text Classification

Deep learning has gained significant attention in natural language processing tasks, including text classification. In this section, we will focus on deep learning-based

models for Bengali text classification. Deep learning models have the ability to automatically learn intricate patterns and representations from text data, making them suitable for complex classification tasks. We will explore here BERT based deep learning architectures.

The proposed DL-based approach has several steps such as 1) Preproccessing 2) Document Representation using word embedding or pre-trained language models 3) Classification

4.2.1 Preprocessing

For deep learning based text classification, we have only used Stopword removal as a part of preprocessing which is already defined in section 4.1.1.

4.2.2 Document Representation using word embedding or pre-trained language models

A) **Word embedding based models**

Word embedding refers to a representation of words in a way that captures their distributional properties. In this representation, each word is mapped to a shared low-dimensional space and associated with a d-dimensional vector. Unlike many other word embeddings, fastText takes into account the morphology of words. It achieves this by using continuous skip-grams, where each word is represented as a collection of character n-grams. For example, if we consider $n = 3$, the word "marvellous" would be represented as "<mar, arv, rve, vel, ell, llo, lou, ous>". This approach preserves subword information and enables the computation of valid word embeddings for out-of-vocabulary words. Hence, fastText can generate vectors for words that were not encountered during the training of word embeddings.

The learning process of word representations in fastText is based on the continuous skip-gram model introduced by Mikolov. This model is simple and performs well even with limited training data. However, it overlooks the internal structure of words. To address this limitation, the creators of fastText propose different scoring functions that take into account the subword information.

Given a word w, the set of n-grams appearing in w is denoted as N_w, where N

is the dictionary size of n-grams. Each n-gram n is assigned a vector representation Z_g. The derived scoring function is then defined as:

$$s(w, c) = \sum_{n \in N_w} Z_g^T V_c \tag{4.2}$$

here, c = context word, V_c = context vector

B) **Pretrained Language Models**

To adapt the transformer model discussed in section 3.2, for the classification task, we incorporate a special start of sequence (SOS) token[1] at the beginning and a special end of sequence (EOS) token at the end.

Furthermore, our experimentation involves four models belonging to two model classes: i) *Multilingual BERT* and ii) *XLM-RoBERTa*.

i) **Multilingual BERT**- BERT is a language model that learns distributed representations from unlabeled texts by considering both the left and right contexts of a token. It utilizes the encoder component of the Transformer architecture. During the language model pretraining phase, two objective functions are employed. The first one is the masked language model (MLM), which randomly masks a certain percentage (15%) of input tokens, aiming to predict the original token's vocabulary ID in that position. The bidirectional nature of BERT allows it to effectively utilize both preceding and succeeding tokens for this task. The second objective is the next sentence prediction (NSP) task, which involves binary classification to predict if two sentences are consecutive in the original text. Positive sentence pairs are created by selecting consecutive sentences, while negative sentence pairs are formed by combining sentences from different documents.

The multilingual variant of BERT (mBERT) is trained on a diverse set of over 100 languages using the largest Wikipedia corpus. Since the number of Wikipedia entries

[1]For BERT the SOS and EOS tokens are respectively [CLS] an [SEP]. For XLM-RoBERTa these are <s> and </s>

varies across languages, data sampling is performed with an exponentially smoothed weighting (with a factor of 0.7). This ensures that high-resource languages like English are under-sampled compared to low-resource languages. Additionally, word counts are sampled in a similar manner to ensure that low-resource languages have an adequate number of words in the vocabulary.

ii) **XLM-RoBERTa**- RoBERTa, an improved version of BERT, enhances its performance by training on larger datasets, utilizing a larger vocabulary, and incorporating longer sequences and larger batches during training. In this improvement, the next sentence prediction (NSP) task is removed, and only the masked language model (MLM) loss is utilized for pretraining.

XLM-RoBERTa, on the other hand, is the multilingual variant of RoBERTa, which is trained using a multilingual MLM. It is trained on a vast amount of data from over one hundred languages, specifically more than two terabytes of filtered Common Crawl data. XLM-RoBERTa has demonstrated impressive performance across various multilingual natural language processing tasks and can achieve comparable results to monolingual language models.

To address the input length constraint of BERT, we have developed a segmentation algorithm which is discussed in **Algorithm** 2, that allows us to process documents with token lengths exceeding the limit of 512 tokens. In our dataset, we observed that a significant number of documents contain more than 512 tokens, making it necessary to employ this approach. The segmentation algorithm involves dividing the long documents into smaller segments using a sliding window technique. We set the window size to 512 tokens, which is the maximum length accepted by BERT, and use a stride of 256 tokens. By applying this sliding window approach, we ensure that we cover the entire length of the document while staying within the allowable token limit. For documents that can fit within the 512-token limit, we directly provide them as input to BERT. However, for those documents that exceed this limit, we divide them into overlapping segments using the sliding window. Each segment is then separately processed by BERT, and the resulting word embeddings are collected. To obtain the final representation of word embeddings for the long doc-

uments, we take the average of the word embeddings obtained from the segmented segments. This averaging step allows us to capture the overall context and meaning of the document despite the segmentation.

Algorithm 2: BERT-based document representation

Input : Text document, d
Output: Document Vector, *doc_vec*

$Tokens \leftarrow$ BERT_Tokenizer(d) ▷ Tokenize the documents
if $|Tokens| \leq 512$ **then**
  $doc\_vec \leftarrow$ BERT_Encoder($Tokens$) ▷ Encode the tokens to a vector
else
  $Segments \leftarrow$ Break_segments(d) ▷ Break document into overlapping segments
  $emb \leftarrow$ zero vector ▷ Initialize a zero vector of length 768
  $i \leftarrow 0$ ▷ Initialize a counter
  for s *in* $Segments$ **do**
    $Tokens \leftarrow$ BERT_Tokenizer(s)
    Seg_vec $\leftarrow$ BERT_Encoder($Tokens$)
    $emb \leftarrow emb +$ Seg_vec
    $i \leftarrow i + 1$
  end for
  $doc\_vec \leftarrow \frac{1}{i} \cdot emb$ ▷ Calculate average vector
end if
return *doc_vec*

By implementing this segmentation algorithm, we can effectively handle documents of varying lengths in our dataset and leverage the power of BERT for natural language processing tasks. The algorithm ensures that even documents with token lengths surpassing the BERT limit can be accurately processed, providing comprehensive and meaningful representations for analysis and downstream tasks.

4.2.3 Classification

To fine-tune the model for specific tasks, we incorporate a task-specific classification head. A general architecture is show in fig 4.4. For the classification task, we focus only on the initial token, corresponding to the start of the sequence token. This token is then utilized to generate the final output probability distribution across the different categories. The hidden representation derived from the start of sequence token can serve as the sentence embedding, which can then be utilized for classification purposes. This results in an embedding vector of dimension H. We have introduced a classification layer with a number of neurons equivalent to the documents's class count. We have used follwing classifier approaches mentioned below.

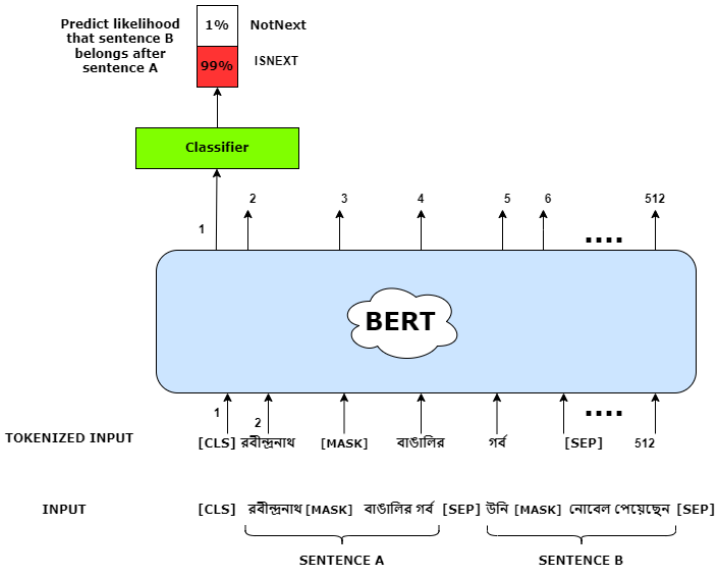
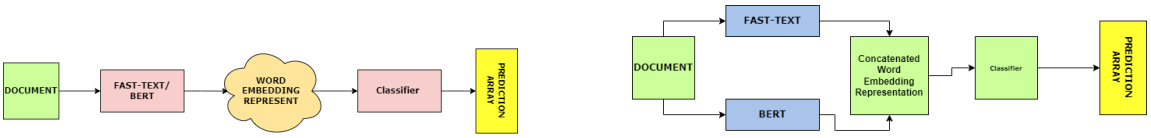


Figure 4.4: General Architecture Of BERT For Text Classification

A. Word Embedding + SVC - Here we have used FastText, BERT and FastText with BERT (which is already discussed in section 4.2.2) for getting the document representation. We have used linear svc as a classifier here to predict the class labels. The model architectures are shown in fig 4.5 where we have used FastText/BERT and Both.



((a)) Word Embedding Model 1 ((b)) Word Embedding Model 2

Figure 4.5: Word Embedding Based Models

B. BERT Based Classifiers - In this setup we have used only BERT models for getting the document representation. We have used 4 BERT models namely 1) Multi-Lingual BERT 2) Multi-Lingual Distill BERT 3) XLM-RoBERTa-Base 4) XLM-RoBERTa-Large and and we have used *TFBertForSequenceClassification* and *TFXLMRobertaForSequenceClassification* as classification layer.

C. Ensemble Of BERT Based Classifiers - In this setup we have done the ensembling of above discussed 4 models and we have suitably chosen the best base classifiers from those 4 models using **Base Classifier Selection Algorithm Using Backward Elimination Method 1** and have created our ensemble-deep learning model using these 2 models namely Multi-Lingual BERT and XLM-RoBERTa-Large. We have shown the accuracy and Weighted F1 score for these models in the subsequent result section.

Chapter 5

Experiments, Evaluations & Results

In this section we will discuss about the training procedure and parameter tuning process for our machine and deep learning models.

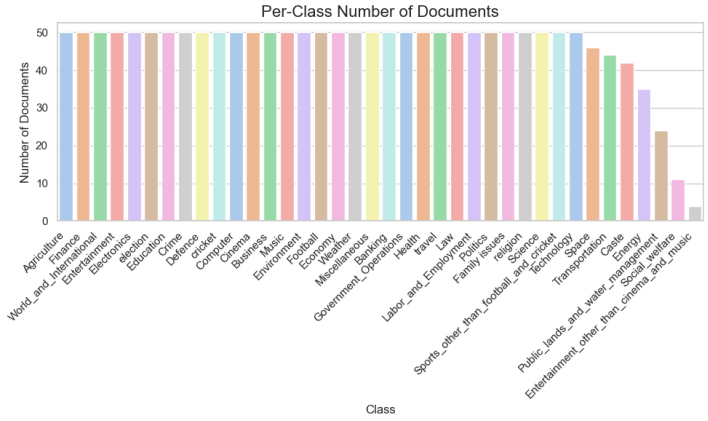
5.1 Description of Dataset

We have done our experiments with two datsets. One dataset is created by us. For second dataset we have used News Classification Dataset[cite]. We have named these datasets respectively as Dataset-1 and Dataset-2.

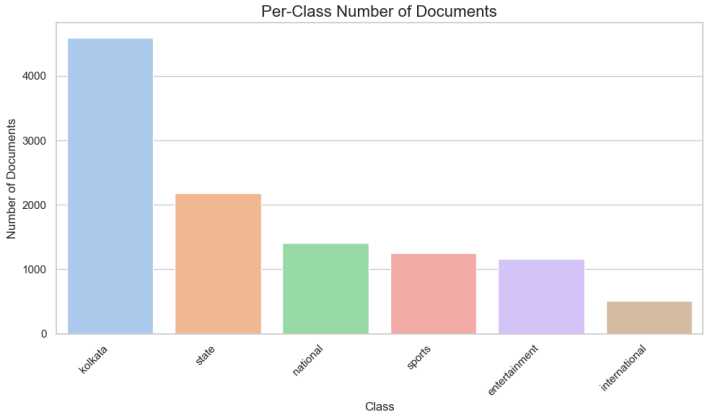
Dataset-1 is randomly splited into train and test set in a ratio of 80:20. This dataset contains 38 classes namely 'Agriculture', 'Banking', 'Business', 'Caste', 'Cinema', 'Computer', 'cricket', 'Crime', 'Defence', 'Economy', 'Education', 'election', 'Electronics', 'Energy', 'Entertainment','Entertainment_other_than_cinema_and_music', 'Environment', 'Family issues', 'Finance', 'Football','Government_Operations','Health', 'Labor_and_Employment', 'Law','Miscellaneous' ,'Music', 'Politics','Public_lands_and_water_management','religion', 'Science', 'Social_welfare','Space', 'Sports _other_than_football_and_cricket', 'Technology', 'Transportation', 'travel', 'Weather', 'World _and_International'. This dataset contains a total of 1756 documents. Each class wise doc-

ument distribution is shown in figure 5.1(a).

Dataset-2 is a publicly available dataset [17]. The dataset is divided into two folders Train & Test. This dataset has 6 classes namely 'entertainment', 'state', 'sports', 'national', 'kolkata', and 'international'. This dataset contains a total of 12,695 documents. Training set contains 11284 documents where as Test set contains a 1411 documents. Each class wise document distribution is shown in figure 5.1(b).



((a)) Documents per class for **Dataset-1**



((b)) Documents per class for **Dataset-2**

Figure 5.1: Description Of Dataset

5.2 Evaluation Metrices

We have used accuracy score and F1 score for evaluating classification performance. Both of these have been calculated using the Scikit-learn package

The accuracy-score function computes the accuracy, either the fraction (default) or the count (normalize=False) of correct predictions. In multilabel classification, the function returns the subset accuracy. If the entire set of predicted labels for a sample strictly match with the true set of labels, then the subset accuracy is 1.0; otherwise it is 0.0. If $\hat{y}_i$ is the predicted value of the i-th sample and y_i is the corresponding true value, then the fraction of correct predictions over n_{samples} is defined as

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1\left(\hat{y}_i = y_i\right)$$

where $1(x)$ is the indicator function.

The $F1$ score can be interpreted as a harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is:

$$F1 = 2 \cdot \frac{(precision \cdot recall)}{(precision + recall)}$$

The weighted F1 score extends this concept to multi-class classification problems. It calculates the F1 score for each class individually and then computes the average, taking into account the class imbalance. Weighted F1 score assigns weights to each class based on their proportion in the dataset. The weight is determined by the number of instances belonging to each class relative to the total number of instances. This ensures that the F1 score is calculated with consideration for the class distribution, giving more importance to the performance on classes with a larger number of instances. By incorporating class weights, the weighted F1 score provides a more comprehensive evaluation of a model's performance in situations where some classes may be underrepresented or have higher importance than others. It takes into

account the overall performance across all classes, while giving more weight to the classes with a larger presence in the dataset. Weighted F1 score is particularly useful when evaluating models on imbalanced datasets, where the number of instances in different classes varies significantly. It provides a more accurate assessment of the model's effectiveness in handling class imbalance and can help identify potential issues with misclassification of minority classes. When interpreting the weighted F1 score, a higher value indicates better overall performance, considering both precision and recall. However, it is important to interpret the results in the context of the specific problem and consider other evaluation metrics and domain-specific considerations for a comprehensive assessment of the model's performance.

5.3 Experiments

We have trained our base classifiers on the above mentioned two datasets. We have seperately tuned each model using GridsearchCV module of sklearn. We have used 3 ensemble technique in this study. For each of the ensembles we have used our base classifiers with appropiate hyper-paramters. We have achieved best result in this setup described in table 5.1 and 5.2

Table 5.1: Hyper-parameters for ML Models for Dataset-1

Base Classifiers & Ensemble Model	Parameters
SVC	C=100, kernel= 'rbf',gamma = 0.001
Random Forest Classifier	n-estimators = 500, min samples split = 1
Logistic Regression	penalty = $l2$, C = 5.0, solver='newton-cg'
Weighted Ensemble	alpha = [0.1]

Table 5.2: Hyper-parameters for ML Models for Dataset-2

Base Classifiers & Ensemble Model	Parameters
SVC	C=1, kernel= 'linear',gamma = 0.01
Random Forest Classifier	n-estimators = 100, min samples split = 5
Logistic Regression	penalty = $l2$, C = 5.0, solver='newton-cg'
Weighted Ensemble	alpha = [0.1]

We have trained[1] the deep learning models using cross entropy loss criterion and Adam optimization algorithm. All models were trained for 10 epochs and 50 epochs for Dataset-1 and Dataset-2 respectively with learning rate : $2e^{-5}$, seed: 1000 and Batch Size : 64. All model parameters are fine-tuned during training i.e, no layer is kept frozen. The model with the best validation set performance is evaluated on the test dataset.

The performance of each developed model is measured in terms of accuracy and weighted F1 score. For model evaluation and comparison, 5-fold cross-validation performance has been considered. In the subsequent sub-section, for each dataset, we will separately discuss the performance of each individual base classifier and the ensemble models created with varying classifier combination strategies and best base classifiers.

5.4 Results

We have tested the proposed ML & DL models on two datasets namely Dataset-1 & Dataset-2. The obtained result on these datsets are discussed in this section.

Result On Dataset-1 - The best results obtained for Dataset-1 is shown in 5.3. In Dataset-1, the model "FastText Averaging Ensemble (Linear Svc + RF)" achieved the highest accuracy of 0.9835 and the highest F1-score of 0.9847. This model outperformed other models in terms of both accuracy and F1-score. The "FastText+BERT+Linear Svc" model, grouped under the "Fusion-Model" category, also showed strong performance with an accuracy of 0.9735 and an F1-score of 0.971.

Result On Dataset-2 - We can see from table 5.3 that, for Dayaset-2, the model "XLM-RoBERTa-Large" attained the highest accuracy of 0.9587 and the highest F1-score of 0.95. The "FastText+BERT+Linear Svc" model, again grouped under the

[1]The models were trained on a single 16 GB NVIDIA Tesla P100 GPU

"Fusion-Model" category, demonstrated competitive performance with an accuracy of 0.7982 and an F1-score of 0.7902.

These results highlight the effectiveness of different models in capturing the underlying patterns and achieving high accuracy and F1-scores. The incorporation of fusion models, such as "FastText+BERT+Linear Svc," also showcased promising results, showcasing the potential of leveraging multiple models for improved performance in text classification tasks

Table 5.3: Model Performance on Dataset-1 and Dataset-2

Dataset	Model	Performance	
		Accuracy	F1-score (weighted)
Dataset-1	FastText with Linear Svc	0.92	0.9219
	FastText Averaging Ensemble (Linear Svc + RF + LR)	**0.9887**	**0.98**
	FastText Weighted Averaging Ensemble (Linear Svc + RF + LR)	0.9716	0.9706
	FastText Stack Ensemble (Linear Svc + RF +LR) with meta classifier as LR	0.9514	0.9484
	TF-IDF with Linear Svc	0.6981	0.6721
	TF-IDF Averaging Ensemble (Linear Svc + RF + LR)	0.7015	0.7095
	TF-IDF Weighted Averaging Ensemble (Linear Svc + RF + LR)	0.6882	0.6802
	TF-IDF Stack Ensemble (Linear Svc + RF + LR) with meta classifier as LR	0.71	0.7153
	Multi-Lingual BERT	0.7118	0.72
	Multi-Lingual DistilBERT	0.6997	0.6943
	XLM-RoBERTa-Base	0.7482	0.7415
	XLM-RoBERTa-Large	0.78	0.7722
	Averaging Deep-Ensemble (Multi-Lingual BERT + XLM-RoBERTa-Large)	0.8001	0.7981
	Fusion-Model (FastText+BERT+Linear Svc)	0.9735	0.971
Dataset-2	FastText with Linear Svc	0.4485	0.4415
	FastText Averaging Ensemble (Linear Svc + RF + LR)	0.4960	0.4875
	FastText Weighted Averaging Ensemble (Linear Svc + RF + LR)	0.5200	0.5204
	FastText Stack Ensemble (Linear Svc + RF + LR) with meta classifier as LR	0.5325	0.5355
	TF-IDF with Linear Svc	0.75	0.75
	TF-IDF Averaging Ensemble (Linear Svc + RF + LR)	0.7591	0.7551
	TF-IDF Weighted Averaging Ensemble (Linear Svc + RF + LR)	0.7701	0.7707
	TF-IDF Stack Ensemble (Linear Svc + RF + LR) with meta classifier as LR	0.7820	0.78
	Multi-Lingual BERT	0.9175	0.9175
	Multi-Lingual DistilBERT	0.8975	0.89
	XLM-RoBERTa-Base	0.9310	0.93
	XLM-RoBERTa-Large	0.9512	0.95
	Averaging Deep-Ensemble (Multi-Lingual BERT + XLM-RoBERTa-Large)	**0.9587**	**0.95**
	Fusion-Model (FastText+BERT+Linear Svc)	0.7982	0.7902

5.5 Comparison With Existing Work

In this section we have given a comparative study between our models and other existing works.

As we have created Dataset-1, there exists no prior results on this dataset. We have used a state of the art ML and DL algorithms for producing the baseline text classification results on this dataset. In this dataset we have got highest accuracy of 98.87% using Fastext embedding and ensemble classifier.

For Dataset-2, we have shown a comparative study in table 5.4 which shows that our proposed model using a single pretrained model(BERT & XLM-ROBERTa) outperform the existing models proposed by T. Alam et al, 2020 [1]. Although our proposed model has some similarity with the model proposed by T. Alam et al, our model differs from the existing model in the way input text is represented. Since the BERT model has an input length restriction(512 tokens), we have used an overlapping window based document segmentation method that breaks a long input document into a collection of contexual segments which are individually represented by BERT. Finally the segment vectors are averaged to obtain the equivalent document vector. But the work presented by T. Alam et al. in their paper did not mention how the longer documents (containing more than 512 tokens) are represented. Most likely, for the above stated reasons, our proposed models performs better than the existing models.

We can also see from table 5.4, that our proposed ensemble model outperforms the existing model by a large margin(i.e 0.0246). The possible reason for achieving better performance by the proposed ensemble model is two-fold : 1) Using overlapping window based document segmentation algorithm for decument representation 2) Applying backward elimination based method for selecting the suitable base classifiers used to build an ensemble.

Table 5.4: Result on Dataset-2(news classification) dataset

	Model	Accuracy	F1 Score
Others	BERT-base [1]	0.9128	0.9128
	XLM-RoBERTa-base [1]	0.9270	0.9284
	XLM-RoBERTa-large [1]	**0.9341**	**0.9340**
Ours	BERT-base	0.9175	0.9175
	XLM-RoBERTa-base	0.9310	0.9310
	XLM-RoBERTa-large	0.9512	0.9512
	Averaging Deep-Ensemble (Multi-Lingual BERT + XLM-RoBERTa-Large)	**0.9587**	**0.9500**

5.6 Error Analysis

In this section, we will discuss about the error analysis of each dataset. We will give an overview about which class is predicted most correctly, and which is less correctly predicted.

For Dataset-1, we have 38 classes which we have discussed in section 5.1.

From this figure 5.2 we can conclude that for most of the classes are predicted very accurately except few classes namely {“Cricket” and “Transportation”, }; {“Entertainment_other_than_cinema_and_music”, and “Entertainment”}.

For Dataset-2, there are 6 classes which we have discussed in section 5.1.

From this figure 5.3 we can conclude that for most of the classes are predicted very accurately except few classes namely {“national” and “state”} ; {“state” and “national”} ; {“kolkata” and “state”}.

We have planned to investigate this part in the future.

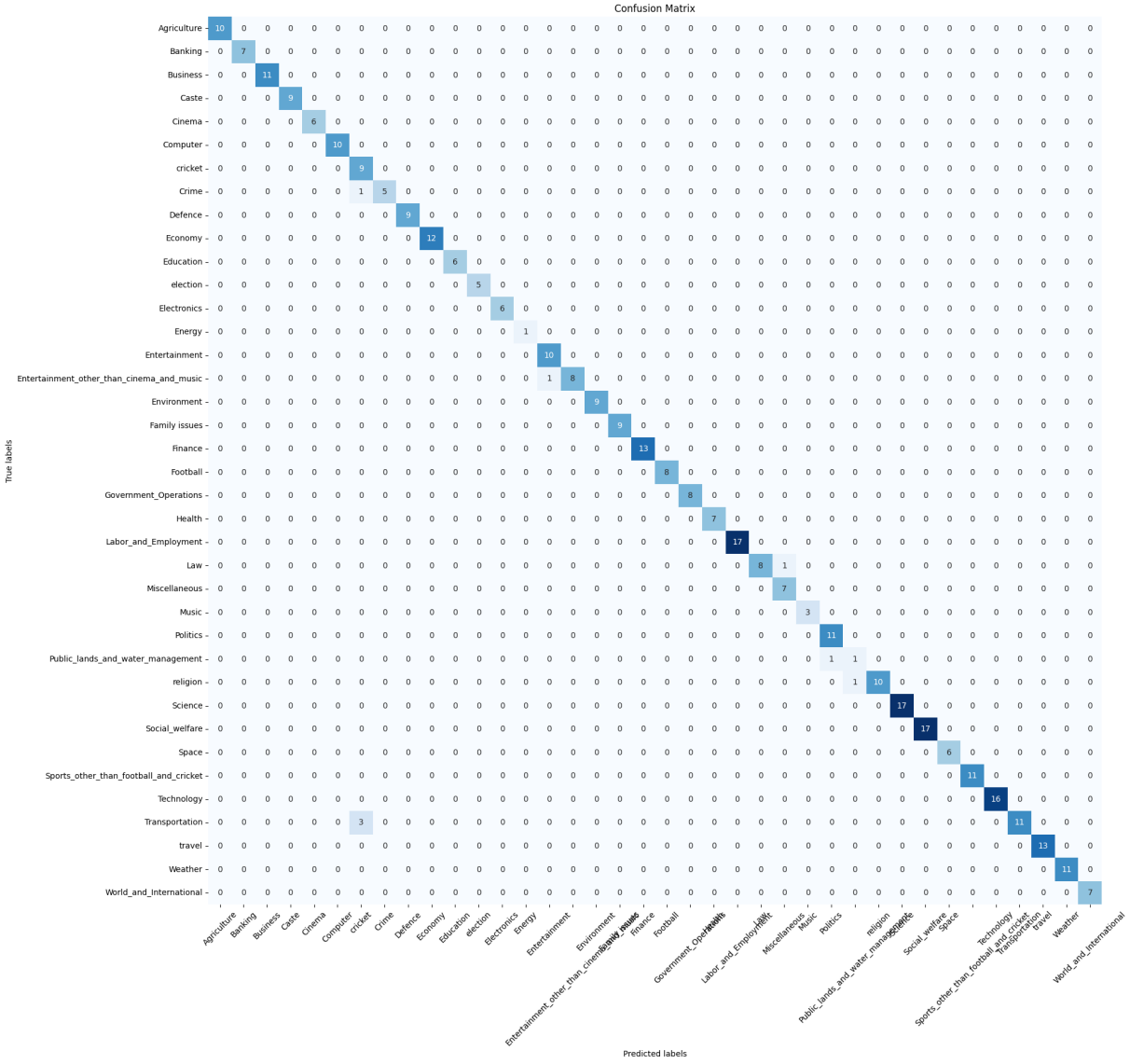


Figure 5.2: Confusion Matrix For Dataset-1

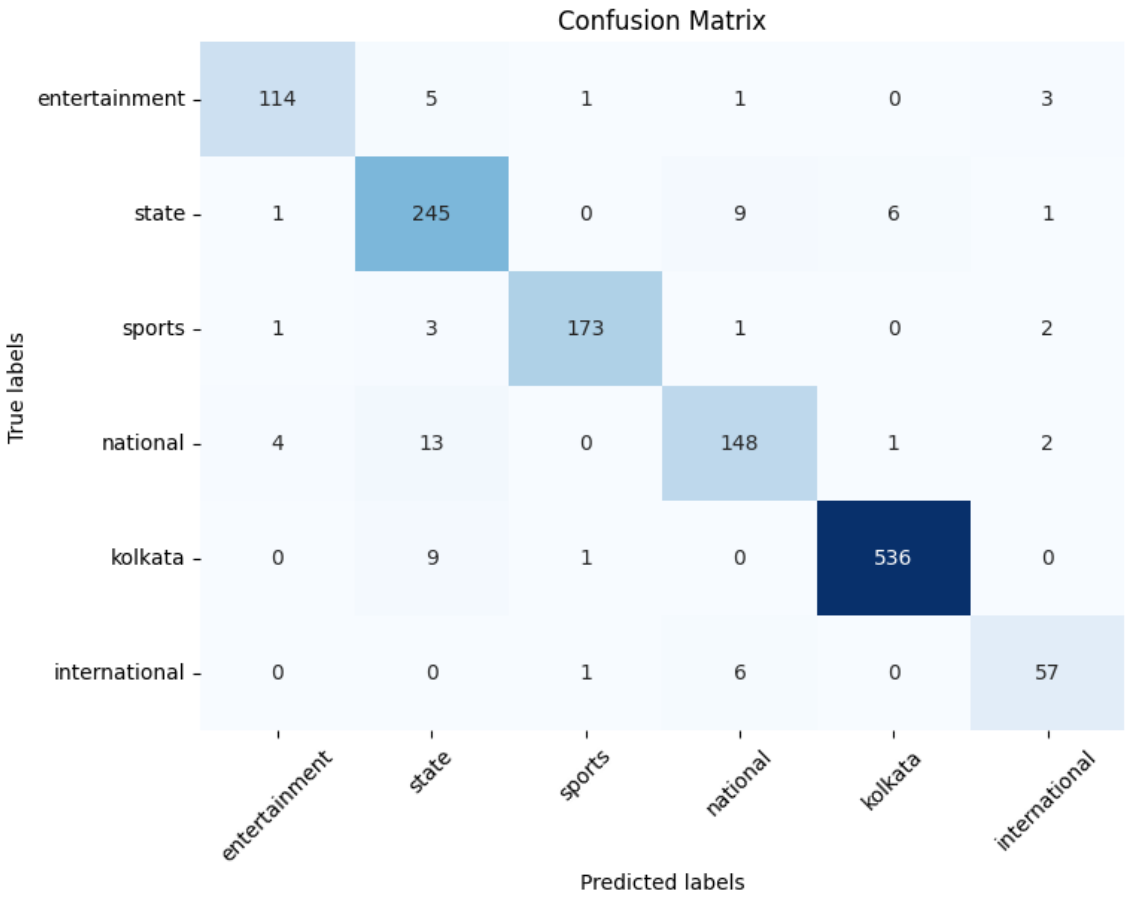


Figure 5.3: Confusion Matrix For Dataset-2

Chapter 6

Implementation

In this section, we provide a detailed overview of the implementation process for our Bengali text classification study, including the selection and tuning of machine learning classifiers, the utilization of pre-trained BERT models for deep learning, and the ensemble construction. The entire experiment was conducted using Google Colab [2] and Kaggle [1] platforms.

6.0.1 Machine Learning Classifiers

We employed various machine learning classifiers from the scikit-learn package [4], including Support Vector Classifier (SVC), Logistic Regression (LR), K-Nearest Neighbors (KNN), Decision Tree (DT), Naive Bayes, and Random Forest. We trained each classifier on our Bengali text classification datasets and evaluated their performance in terms of accuracy.

We have initially fine-tuned each classifier individually using the GridSearchCV method, which systematically explores the hyperparameter space of each classifier to find the optimal combination of hyperparameters. The performance evaluation was conducted by averaging the accuracy scores obtained from 5-fold cross-validation.

To determine the best feasible base classifiers for our ensemble model, we sorted them in descending order based on their accuracy. This allowed us to prioritize the classifiers with higher accuracy for ensemble construction using our proposed

[1] https://www.kaggle.com/

algorithm 1 for base classifier selection.

6.0.2 Deep Learning with BERT

For the deep learning part of our study, we leveraged pre-trained BERT models, which have shown excellent performance in various NLP tasks. We utilized the Hugging Face library [15] to download the pre-trained BERT transformers specifically designed for Bengali text. We have used base and large versions of i)Multilingual BERT and ii) XLM-RoBERTa.

Since BERT models have input length constraints, we applied our segmentation algorithm to bypass this limitation. The algorithm intelligently divided longer Bengali text sequences into smaller segments and utilized the contextual embeddings from BERT to ensure accurate classification.

Similar to the machine learning classifiers, we conducted individual evaluations and fine-tuning of the pre-trained BERT models using the GridSearchCV method. This enabled us to optimize the hyperparameters of the BERT models specifically for Bengali text classification.

6.0.3 Ensemble Construction

To harness the power of both machine learning and deep learning approaches, we constructed a deep ensemble model. Following the same approach used for the machine learning classifiers, we selected the best feasible base classifiers from both the machine learning and deep learning models based on their individual performance.

The ensemble model combined the predictions of the selected base classifiers using our proposed algorithm for base classifier selection. The algorithm utilized the backward elimination method to iteratively eliminate underperforming classifiers, resulting in an optimized ensemble model.

The entire implementation was carried out using the Google Colab and Kaggle platforms, which provided the necessary computational resources and libraries required for our study.

Chapter 7

Conclusion and Future Work

In conclusion, this study has undertaken a comprehensive exploration of Bengali text classification using machine learning and deep learning approaches. The research has contributed to the field by evaluating the efficacy of different multilingual pre-trained models for Bengali text classification tasks and proposing novel algorithms to address challenges related to base classifier selection and input length constraints.

Through extensive evaluation and analysis, the study has identified the most effective pre-trained models for Bengali text classification, providing valuable insights for researchers and practitioners in developing high-performing classifiers for Bengali language data. We have proposed an algorithm for selecting base classifiers used to build an ensemble model. The proposed algorithm for base classifier selection has improved the efficiency and effectiveness of ensemble models by selecting the most relevant and competent base classifiers. Additionally, the algorithm for dealing with the input length constraint of BERT models has enabled the accurate classification of longer Bengali text documents.

Future Work:

i) While this study has made significant contributions to Bengali text classification, there are several avenues for future research and improvement. Some potential areas for future work include:

ii) Fine-tuning pre-trained models: Further exploration can be done to fine-tune the selected pre-trained models specifically for Bengali text classification. Fine-tuning can help adapt the models to the specific linguistic characteristics and nuances of the Bengali language, potentially leading to even better performance.

iii) Incorporating contextual information: The proposed algorithms can be enhanced by incorporating more contextual information into the models. Techniques such as attention mechanisms or contextual embeddings can be explored to capture the context-dependent nature of the Bengali language, improving the classification accuracy.

iv) Handling domain-specific challenges: Future research can focus on addressing domain-specific challenges in Bengali text classification. For example, developing techniques to handle informal language, dialects, or specific industry domains can lead to more accurate and tailored classification models.

v) Evaluation on larger datasets: The performance of the proposed approaches can be further evaluated on larger and diverse datasets. This will provide a more comprehensive assessment of their effectiveness and generalizability across different domains and data distributions.

vi) Exploring other deep learning architectures: While this study focused on BERT models, there are other deep learning architectures that can be explored for Bengali text classification. Investigating the effectiveness of models such as Transformer-XL, GPT, or XLNet on Bengali language data can provide valuable insights and potentially improve classification performance.

By addressing these future research directions, we can continue to advance the field of Bengali text classification, developing more accurate, efficient, and domain-specific models. The outcomes of future work will have practical implications for various applications involving Bengali text, including sentiment analysis, document categorization, and information retrieval. With minor modifications in the proposed models it can applied to text classification task in other indian languages like Hindi, Marathi, Tamil etc.

Bibliography

[1] T. Alam, A. Khan, and F. Alam. Bangla text classification using transformers. *arXiv preprint arXiv:2011.04446*, 2020.

[2] E. Bisong. *Google Colaboratory*, pages 59–64. Apress, Berkeley, CA, 2019.

[3] L. Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[4] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[5] A. Chandra and X. Yao. Evolving hybrid ensembles of learning machines for better generalisation. *Neurocomputing*, 69(7-9):686–700, 2006.

[6] A. N. Chy, M. H. Seddiqui, and S. Das. Bangla news classification using naive bayes classifier. In *16th Int'l Conf. Computer and Information Technology*, pages 366–371. IEEE, 2014.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] A. Dhar, N. S. Dash, and K. Roy. Classification of bangla text documents based on inverse class frequency. In *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*, pages 1–6. IEEE, 2018.

[9] A. Dhar, N. S. Dash, and K. Roy. An innovative method of feature extraction for text classification using part classifier. In *Information, Communication and Computing Technology: Third International Conference, ICICCT 2018, New Delhi, India, May 12, 2018, Revised Selected Papers 3*, pages 131–138. Springer, 2019.

[10] A. Dhar, H. Mukherjee, N. S. Dash, and K. Roy. Performance of classifiers in bangla text categorization. In *2018 International Conference on Innovations in Science, Engineering and Technology (ICISET)*, pages 168–173. IEEE, 2018.

[11] Y. Dong, J.-B. Cordonnier, and A. Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning*, pages 2793–2803. PMLR, 2021.

[12] S. Garg, T. Vu, and A. Moschitti. Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7780–7788, 2020.

[13] F. Haghighi and H. Omranpour. Stacking ensemble model of deep learning and its application to persian/arabic handwritten digits recognition. *Knowledge-Based Systems*, 220:106940, 2021.

[14] M. S. Islam, F. E. M. Jubayer, and S. I. Ahmed. A comparative study on different types of approaches to bengali document categorization. *arXiv preprint arXiv:1701.08694*, 2017.

[15] S. M. Jain. Hugging face. In *Introduction to Transformers for NLP: With the Hugging Face Library and Models to Solve Problems*, pages 51–67. Springer, 2022.

[16] F. Kabir, S. Siddique, M. R. A. Kotwal, and M. N. Huda. Bangla text document categorization using stochastic gradient descent (sgd) classifier. In *2015 International Conference on Cognitive Computing and Information Processing (CCIP)*, pages 1–4. IEEE, 2015.

[17] A. Kunchukuttan, D. Kakwani, S. Golla, A. Bhattacharyya, M. M. Khapra, P. Kumar, et al. Ai4bharat-indicnlp corpus: Monolingual corpora and word embeddings for indic languages. *arXiv preprint arXiv:2005.00085*, 2020.

[18] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[19] Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural networks*, 12(10):1399–1404, 1999.

[20] A. K. Mandal and R. Sen. Supervised learning methods for bangla web document categorization. *arXiv preprint arXiv:1410.2045*, 2014.

[21] A. Mayr, H. Binder, O. Gefeller, and M. Schmid. The evolution of boosting algorithms. *Methods of information in medicine*, 53(06):419–427, 2014.

[22] R. Polikar. Ensemble learning. *Ensemble machine learning: Methods and applications*, pages 1–34, 2012.

[23] L. Rokach. *Ensemble learning: pattern classification using ensemble methods*. World Scientific, 2019.

[24] O. Sagi and L. Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[25] C. Sun, X. Qiu, Y. Xu, and X. Huang. How to fine-tune bert for text classification? In *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*, pages 194–206. Springer, 2019.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[27] Z. Zhang, Y. Wu, H. Zhao, Z. Li, S. Zhang, X. Zhou, and X. Zhou. Semantics-aware bert for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9628–9635, 2020.

[28] Q. Zhu and J. Luo. Generative pre-trained transformer for design concept generation: an exploration. *Proceedings of the Design Society*, 2:1825–1834, 2022.