A study on the efficacy of natural language generation techniques for similar writing personalities

by

Ahmed Anwar
20301077
Hasan Mohammod Tanzir
20101598
Abdul Halim Hosain
20101300
Taslima Islam
20101603
Nusrat Zaman Raya
23241034

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

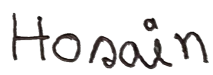
Department of Computer Science and Engineering
Brac University
January 2024

Declaration

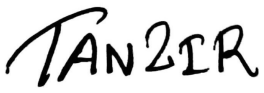
It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

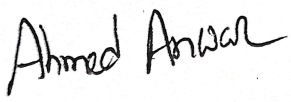


Abdul Halim Hosain
20101300



Hasan Mohammod Tanzir
20101598

Taslima Islam
20101603



Ahmed Anwar
20301077

Nusrat Zaman Raya
23241034

Approval

The thesis titled "A study on the efficacy of natural language generation techniques for similar writing personalities" submitted by

1. Ahmed Anwar (20301077)
2. Hasan Mohammod Tanzir (20101598)
3. Abdul Halim Hosain (20101300)
4. Taslima Islam (20101603)
5. Nusrat Zaman Raya (23241034)

Of fall, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 9, 2024.

Examining Committee:

Supervisor: (Member)

Dr. Farig Yousuf Sadeque
Assistant Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator: (Member)

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
School of Data and Sciences
Brac University

Head of Department: (Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Over the past two decades, the domain of Natural Language Processing has undergone a remarkable transformation enabling machines to generate text, summarize content, paraphrase and analyze sentiment. The captivating idea of analyzing and copying someone's writing style is no longer impossible. Pushing boundaries further, we have embarked on a journey to implement such a model for the Bengali language by utilizing the approach of style transfer through the application of deep learning using LLM. One's writing personality can be identified by training the model by imputing a set of documents (notes, books, etc) authored by the writers only. The system will be able to extract important information to recreate sentences with similar structural properties used by the author. Additionally, it will also be able to detect whether a particular sentence structure synchronizes with that author's distinctive style. The outcome of our model aims to fulfill the need of a particular writing taste of an author, as requested by the user. In essence, our model blends technology and art to write in a way that is reminiscent of their favorite Bengali author. Our proposed model not only skillfully excels in authorship classification and mimicking their style, but also stands resilient against potential adversarial attacks, making it a strong and unyielding system that aligns well with our research objective.

Keywords: Natural Language Processing; Writing personality; Writing style-LSTM; Style Transfer; adversarial attacks; LLM; T5 model; mT5; BanglaT5; BanglaBERT.

Table of Contents

List of Figures

Chapter 1

Introduction

In the realm of ever-evolving AI and Machine Learning, this research paper commences on a groundbreaking journey utilizing NLP as a transformative tool for the advancement of research works in the Bengali language by proposing a model that is able to detect a piece of fake writing from that of what the original author wrote. To do this, we have trained our model in such a way that it'll identify one's writing personality and retrieve crucial properties to recreate sentences with similar structures. To break it down, we started by training our model on a diverse dataset where we've extracted unique patterns that serve as the literary fingerprints of various authors and later guide the classification of texts in various ways. In the classification phase, to effectively categorize and differentiate texts, we've implemented a multifaceted approach employing four distinct methods. These approaches are underpinned by the utilization of bidirectional LSTM and BanglaBERT. First, the model makes a comparison between texts written by Rabindranath Tagore and those by other authors, spanning both historical eras (Rabindranath and contemporary era). Secondly, it distinguishes between the writing of the authors of the Rabindranath era, such as - Begum Rokeya, Sarat Chandra Chattopaddhay, Bankim Chandra, and contemporary authors, including- Humayun Ahmed, Zafor Iqbal, and Ahmed Sofa. Then, the model extends its exploration to a broader spectrum by detecting distinctive writing styles from multiple authors that ensure applicability to a diverse range of authorship scenarios. The fourth and final approach in classification addresses preserving adversarial attacks. This crucial aspect makes sure the model's robustness, where the model can defend itself and preserve itself against potential misuse. In this classification phase, our model is equipped to separate the authenticated writing from the generated one. By combining these four approaches, our classification phase performs efficiently and precisely in the realm of literature. Moving on to the text generation part, we've fine-tuned two models, BanglaT5 and mT5. Here, from the patterns obtained by the chosen author, Rabindranath Tagore, we're generating his writing style anew. Not to mention, both of the tasks of identifying an author and generating their style are carried out at the sentence level. The driving factor behind our decision for this research is from the realization of the fact of insufficiency of advancements specific to the Bangla language. Existing methodologies in this field often lack efficiency and suitability for Bangla, which motivates us to explore some distinctive approaches which will be explained throughout this paper.

Chapter 2

Research

2.1 CHALLENGE OVERVIEW

In the big world of words, the English language has been a wealth of research that features extensive scholarly works, comprehensive datasets, models, relevant resources, and so on. Now many NLP enthusiasts and researchers eagerly dedicate themselves to match this literary quest. However, it's ironic that the Bangla language is still left with a void as it lacks relevant versatile models and essential resources that have fallen behind the times in the Bengali landscape. This gap presents an unfortunate opportunity for deceptive practices. As perpetrators are flooding online bookstores with fake books, they falsely claim to be written by authentic authors. They replicate the original writing style so well that it is too hard for Bengali emotional individuals to distinguish between the authentic and the fake version. That's where our research comes in handy, which is dedicated to safeguarding the authenticity of original writings and exposing the impostors. To achieve this objective, we set upon two sequential steps of adventure. Initially, we immerse ourselves in the extraction of the unique styles of targeted authors, fetching their essence from an abundance of data. Second, building up this knowledge, we're venturing into the text generation step. Incorporating style transfer techniques, we then imitate their signature patterns to create a distinctive context that sets the real apart from the fake. In our thesis, our focus extends beyond the mere identification of authorship. To sum up, we'll classify authorship by extracting the identifiable attributes and writing patterns. We'll bring their styles back to life using the insights we extract. Ultimately, we're on a quest to separate the genuine authors from the disguised ones. Thereby, we try to establish a basis for a defense against adversarial attacks and the future scope of research in this genre in the Bangla Language.

2.1.1 People affected by this problem

CURRENT BENGALI AUTHORS & RESEARCHERS

Khatun (2022) highlighted the necessity of Bangla-style transfer techniques and author identification in the lives of current Bengali authors and scholars. In accordance with that, our proposed model helps to detect and identify the input writing by comparing its writing pattern with that of an author's writing from the dictionary. This way current Bengali authors can be benefitted as our system is able to protect their writings from adversarial attacks, and dedicated computational & forensic linguistic

researchers are able to develop smarter techniques with the help of our model that can tackle issues such as diacritics and ligatures, resulting in more accurate author identification.

CONTENT ANALYSTS AND CYBERSECURITY COMPANIES

In the intricate world of Bangla typing style, a hidden challenge awaits content analysis, cybersecurity companies, and digital forensics firms. The organizations encounter the obstacle of identifying anonymous or deceitful content authors in the Bangla language due to the complexity and variability of typing styles, linguistic nuances, and the use of local colloquialisms. The sheer diversity in Bangla script and writing conventions further complicates the task, making it difficult to attribute content to specific individuals or groups. Our proposed model can assist in this regard by providing a context of narrative-style understanding, author-specific wording, and phraseology in Bangla text, helping organizations identify suspicious or potentially harmful content.

PLAGIARISM DETECTION PLATFORMS

There are many platforms such as educational institutions, content creators, and plagiarism detection applications, that can get benefit from the proposed system to find out cases of plagiarism in the context of Bangla. By detecting the writing style of an author, it becomes easier to unmask situations of academic dishonesty or copyright violation.

BANGLA LANGUAGE USER

Bengali writers, bloggers, journalists, and social media users all are impacted by the issue. By deterring anonymity and assuring responsibility, trustworthy author identification systems in Bangla can aid in the protection of their intellectual property rights, the fight against identity theft, and the promotion of responsible digital conduct.

MASS COMMUNITY

By encouraging a safer online environment in the Bangla language, the issue has an indirect effect on the wider public. For Bangla language users, an accurate author identification system can ensure a more dependable and trustworthy online experience by discouraging harmful activities like disseminating false information, hate speech, or counterfeit content

2.1.2 Attempts that have been made to solve the problem

So far, different writing style detection models have been built in English for instance: Detection of changes in literary writing style using N-grams as style markers and supervised machine learning [24], Writing style change detection on multi-author documents [20], etc. The closest research work that is relevant to our problem is- Effective writing style transfer via combinatorial paraphrasing [9]. It can generate texts for specific authors/personalities but unfortunately, it is also applicable to the English language only. On the other hand, for Bengali language we were able to find

a research work for the purpose of author detection in particular named "Authorship Attribution in Bangla Literature (AABL) via Transfer Learning using ULMFiT" [11] and some text generation models such as, Context-driven Bengali Text Generation using Conditional Language Model [19], A Bengali Text Generation Approach in Context of Abstractive Text Summarization Using RNN [8], Bengali Text generation Using Bi-directional RNN [7], etc. But there were no relevant research works on author-specific text generation models and adversarial attack detection for Bangla even though it is the 7th most spoken language in the world [10], which subsequently ensued the idea or stipulation of building such a model to ensure proficiency of future research works.

2.1.3 Complexities that will arise if the problem is not solved

As we have already discussed above, all style transfer techniques that use NLP for accurate text imitation and authorship attribution are based on the English Language, and only a handful of work has been done in Bangla Literature. So, if the problem is not addressed nor solved, we might never be able to achieve strong style transfer performance. Also, without accurate content authentication and authorship classification, there is a risk of misattributing content, which unfortunately might lead to confusion about the true origins of literary works. It can risk the privacy of the original wordsmith [9]. If we are unable to execute our model, we won't have an implemented tool to defend against adversarial attacks. Moreover, where generalized paraphrasing tools are widely available, we won't have access to a personalized paraphrasing tool. This model is crucial for legal documentation and content creation. Without it, we won't be able to assist authors in distinguishing between fake and real identifications. Analyzing major changes between the works of authors from different eras, such as computational analysis won't be introduced either. And, thereby, these are the reasons why such a project is invaluable and must be solved.

2.2 RESEARCH OBJECTIVES

This study aims to use deep learning to detect a specific writing style of different authors from a given input, and extract symbolic features of their written work, and further employ those retrieved information to generate documents that skillfully mimic their signature pattern. On top of that, we're also on a mission to implement techniques to distinguish between authentic and fake documents by protecting them from unwanted adversarial attacks. The main objectives of our research are:

- Building a quality dataset and a model that can classify texts authored by Rabindranath Tagore and those by other authors (from both the Rabindranath era and the current era).
- Establish a dataset and a model that includes texts from the Rabindranath Tagore era and the authors from the current era and distinguish between them.
- Compiling a quality dataset that encompasses writings from multiple authors, beyond just Rabindranath Tagore. This extends the model's applicability to handle a diverse range of authorship scenarios.

- Implement mechanisms within the model to resist adversarial attacks.
- Enable the model to extract unique patterns (pattern recognition) from the writing styles of different authors.
- Generation of texts in Bengali that would be unique for every person based on their personality in the future.
- Protection from unidentified plagiarism allegations and copyright infringement. This also helps us to get protection from unwanted adversarial attacks.
- Paving the way for state-of-the-art plagiarism detection models to authenticate and preserve safe content in literature, preventing plagiarism.
- While generalized paraphrasing tools are widely available on online platforms, they often come with privacy concerns. Therefore, our mission goes beyond these general tools as we aim to provide access to a personalized paraphrasing tool.
- Enhancing scopes of future prospects, including building models with a view to encouraging more research work on the Bengali language.

Chapter 3

LITERATURE REVIEW

As the dependency on online-based work submission and publication has escalated over the past years, most submissions and published works seek the hand of plagiarism. To avoid such illegal activities, we have decided to work on a project called "Bangladigm" that initially aims to identify similar Bangla writing styles (in our case, only that of Rabindranath Tagore) by matching the input sentence to an existing dataset consisting of colossal Bangla words of a specific author and in return, giving us the name of the original author according to their writing style. And in the next phase, it is able to generate natural sentences in accordance with the input sentence based on the trained writing style. This helps us in many ways, as discussed above.

To have a better understanding of Deep Learning, Natural Language Processing and State-of-the-art (SOTA) models to design such a style transfer technique, we had to read quite a bit of literature from the internet related to our topic on a scholarly search engine, Google Scholar. The topics that we came across that were concurrent to our project design were combinatorial paraphrasing, authorship attribution, tokenization, POS tagging, lemmatization, spell checking, etc. These keywords redirected us to papers that examined various techniques to give a better and more accurate model for style transfer. After searching for many papers, we have come across only a very few that helped us envision a proper architecture for our model. So, for further filtration, we decided to use only those peer-reviewed research papers that had a numerous number of citations on it. There are 15 related papers that the five of us have analyzed and tried to incorporate with our own thesis paper. They are reviewed briefly below, along the summary of their aim of research, methodologies applied, evaluation, and a few limitations that should give a clear idea of what we are trying to do. The review papers are:

A paper authored by Aisha Khatun, Anisur Rahman, Md Saiful Islam, Hemayet Ahmed Chowdhury, and Ayesha Tasnim, from the Department of Computer Science and Engineering, Shahjalal University of Science and Technology [11] suggested to apply the ULMFiT architecture for authorship attribution on Bangla Literature, since this field of language in NLP has still been untouched by most researchers. The definition of Authorship Attribution is to recognize the original author of an input text by using NLP and computational linguistics to conserve the Bangla Literary heritage. So, the authors of this paper introduce a kind of state-of-the-art

transfer learning technique, called the ULMFiT model (Universal Language Model Fine-tuning) that does fine-tuning on a compact dataset after being trained off of a macroscaled language model. This allows the model to vigorously learn multiple representations of the Bangla language and process them accordingly even with inadequate, unlabelled data. Similar to our thesis, the authors of this paper create a dataset from a Bangla literature corpus that is incorporated with a massive amount of works of multiple authors in different time frames, genres, etc. The dataset is called BAAD16 (this is the largest dataset in Bangla literature for authorship attribution containing 16 authors and the number of words available per author and the total number of words, 13.4+ million) [11] and BAAD6 (which is a 6 Author dataset collected and analyzed by Hemayet et al. [5]. This dataset's total number of words and unique words are 2,304,338 and 230,075 respectively). The methodology of preprocessing the dataset starts by the tokenization of words and sending the processed data to the UMLFiT architecture for further training by learning the patterns and specific key linguistic features of every data of different authors. This way, the model is able to successfully predict the different authors of a given input text. To evaluate the performance of the proposed project of this paper, the authors performed several experiments and comparisons with other baseline methods like Naive Bayes and Support Vector Machine (SVM) to establish the fact that the ULMFiT architecture has a higher value for recall, accuracy and precision in case of Bangla literature. It also talks about some elements such as the size of the training dataset, the number of authors, the presence of noise in the text, effect of pre-training the dataset, which all affect the performance of the proposed model. One other method the authors highlight is called anonymization of data. In this process, the private and personal information of authors is protected to establish the ethical use of all authored attribution techniques. The authors also provide a perspicacious analysis of the limitations and future improvements such as the incompetences created due to the selection of a specific corpus, the opportunity of transferring the proposed approach to other languages or multilingual settings, the difficulties in anonymization of data, and the consequences of the train dataset size and quality to ensure better accuracy of the model. So finally, after reading this paper we can apply some of its techniques to create our model.

A research paper authored by Tommi Grondahl and N. Asokan [9] discusses how writing style transfer using combination can be implemented. Writing style transfer refers to the task of autonomously transforming a given input text into another text in accordance with a desired writing style. Paraphrasing refers to the transformation of a text while keeping the gest of the original text. According to the authors, all existing methods are unsuccessful in conserving the semantic linguistic elements of the output text while implementing the style transfer techniques to it, thus, they proposed a combinatorial model. Just like normal mathematical combinatorial, the task of this model is to generate multiform text sentences by applying a number of linguistic rules to a particular input text and selecting that output sentence with a higher probability of matching the gest of the original one through a double-step procedure. Initially, the system's work methodology includes following a rule-based approach the "candidate generation step" is carried out by generating multiple paraphrased sentences and secondly, a trained classifier labels the generated sentences from step 1 according to their stylistics and fluency to select an appropriate can-

didate called the "candidate selection step". The dataset used for this thesis is a standard wide-ranged data corpus with various writing styles including 245691 paraphrase pairs. It also contained four two-class corpora for sentence-based datasets and has 7899 + 7270 + 6766 additional training sentences for multiclass. It also has a BrennanGreenstadt corpus (BG) consisting data of 12 authors and the Extended-BrennanGreenstadt corpus (EBG) consisting data of 45 authors. Furthermore, the authors evaluate the accuracy of their model through experiments including comparisons between the results obtained from their model versus the result of other existing style transfer models. They also repeat the process for two distinct datasets to ensure the metrics. Some processes such as content preservation, fluency, and style accuracy, are taken into account like calculating the performance metric of this model. The authors also shed light on the limitations and future possibilities/improvements of their proposed system, including the processing issue of longer sentences that should follow a similar writing style and the incompetence of a rule-based paraphrasing technique would bring to ensure the quality of a writing. Future possibilities include the advancement of research in this field with slight variations such as using neural network-based paraphrasing models to accomplish the task and also include how it could serve various domains such as NLP, creative writing and computational linguistics, etc. So, by following these techniques, a style can be transferred via. a text which is also the main target of our thesis.

Another paper authored by Kevin Lin, Ming-Yu Liu, Ming-Ting Sun, and Jan Kautz [14], uses style transfer to generate multiple outputs for a single given input text. The methodology of data processing involves the feeding of data into a Generative Adversarial Network (GANs) framework that generates multiple outputs through two main components, a style encoder and a style decoder. The task of a style encoder is to imprint the desired style onto the input text by mapping the style onto the data, whereas, the style decoder generates several output sentences with the captured essence of the desired writing style we want by preserving its semantic content and linguistics. The process of training the model includes the introduction of a style diversity loss function that ensures diversity based on style amongst the output sentences and the integration of a style reconstruction loss to ensure the fluency of the implementation of the desired style on the input text data. Even though this is an effective approach, it still has limitations, such as identifying the fact that multiple and distinct text generation using NLP is quite a tough task to accomplish. The authors also discuss its future possibilities and improvements, such as the study of GANs to learn accurate but various representations of the same data through generative modeling, which would strike an interest in other researchers for further study on this particular field. The authors have also shed light on the evaluation process to find its performance metrics, where product review datasets from Amazon contain 277, 228 positive and 277, 769 negative review sentences for training and 500 positive and 500 negative review sentences for testing. The length of a sentence ranges from 8 to 25 words) [14] and Yelp (contains a training set of 267, 314 positive and 176, 787 negative sentences, and a test set of 76, 392 positive and 50, 278 negative testing sentences. The length of a sentence ranges from 1 to 15 words.) [14] are used to train and learn their model. The process of experimental implementation for obtaining a satisfactory performance metric includes the tracking of style accuracy, diversity of generated outputs, semantic content conservation,

etc. and comparison of the results with baseline methods to prove the computational precision of their suggested model. So, finally, the authors have discussed the contribution of this paper on the field of NLP. It allows researchers to investigate different approaches to control the degree of style transfer on an input sentence. It also allows researchers to try integrating a wider and balanced dataset to improve the quality and relevance of the output. Finally, this model could be tested on a larger domain and other languages, such as Bangla (which helps us reach our goal), etc. So, this research paper has helped us gain appropriate and extensive knowledge about the style transfer technique, which will be needed by us to create a style transfer approach of our own that is compatible with the Bangla language in our thesis.

Furthermore, a paper authored by Keith Carlson, Allen Riddell, and Daniel Rockmore [18] discusses the issues of text style transfer techniques. The methodology of processing the data begins with an unsupervised approach for the separation of style and content in the input text by using content embeddings. The process is a double-step approach, where initially an unsupervised clustering algorithm aims to detect groups of different styles within the dataset, called the style extraction step. It is followed by the learning of content embeddings by the utilization of denoising autoencoders [18]. Finally, a method named style exaggeration tends to change the content embedding while effectively conserving the style information of the original text. To ensure the conservation of style information, the authors also formulate a reconstruction loss function. Just like the previous paper, this paper also contributes to the field of text style transfer, which allows more and more researchers to work on this comparatively new field of study. This model also contributes to the robust flexibility of style transfer which can be seen through performance metric experiments. So, to evaluate the performance metric of this model, again product review datasets from websites such as Amazon and Yelp are used to train and learn their model just like the above-mentioned thesis paper, but unfortunately, the details of the number of data used are vague. This paper also has limitations and future improvements that are very similar to that of the paper reviewed just before this. So, by reading this paper we now have a clearer idea of style separation by using content embeddings which will be helpful for our thesis.

Another paper authored by Jad Kabbara and Jackie Chi Kit Cheung from the School of Computer Science, McGill University [2], has discussed the efficacy of natural language generation processes by using style transfer approaches. This is done by the utilization of recurrent neural networks (RNNs). The methodology suggested by the authors to make this system work, includes the usage of long short-term memory (LSTM) network which is a neural network architecture that can conduct style transfer technique. They do this by capturing the desired writing style attributes by using a style embedding approach which is integrated into the next step, the generation phase. The LSTM architecture based on encoder-decoder structure is trained by using a large data set of text corpus which pushes the model to learn the semantic and statistical patterns of a writing style. It also learns the style specific characteristics of various writing styles. The datasets include: the complete works of Shakespeare, the Wikipedia Kaggle dataset, the Oxford Text Archive (literary texts), and Twitter data. But no detail of the number of data has been found in

this paper. To get the desired style transfer, the learned style embeddings are mixed with the content embeddings of the input text, and this process is called style interpolation, where the model is able to generate one text that expresses a mixture of desired writing styles by manipulating the interpolation ratio. The opportunities and contributions of this paper includes study of natural language generation by using RNN which allows for a fine-grained control on the perseverance of stylistic elements of a paraphrased text. The evaluation of this model's performance metric is done by the execution of both automatic and human experiments, which demonstrates the robustness of this methodology. Future opportunities discussed by the authors include the leverage of more studies in this area of statistics transfer. The integration of additional linguistic attributes and context information, such as contextual embeddings and attention mechanisms can be done to enhance the quality of this model. Another future opportunity of this research would be to build its transferability to a wider range of domains and languages, for multilingual utility. The insights provided by this paper is an important factor to proceed in our field of work.

Now, we'll see a paper authored by Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, Rui Yan [6], researched about the style transfer of text with non-parallel data and how it is not up to the mark because of insufficient parallel data and good evaluation metrics for evaluating the models. To reduce these problems, the authors of this paper proposed two models which are: multi-decoder and style-embedding. These two models are related to the neural sequence-to-sequence model. In this paper, the authors have analyzed different models but the best outcome comes from multi-decoder and style-embedding models which are verified by a number of experiments conducted to figure out its performance metrics. Furthermore, the two evaluation matrices that have been introduced in this paper for measuring the model efficiency are transfer strength and content preservation. These two metrics are working better than other metrics. Moreover, two datasets have been used for the research in this paper which is called a non parallel corpora. The name of the dataset are paper-news title dataset, called the positive-negative review dataset. The amount of the test data was 2000 sentences and the remaining was for the training data.

In another research paper that was authored by Se Won Jang, Jesik Min, Mark Kwon titled [4], it proposed two models for style transfer. Those are simple sequence-to-sequence models with attraction and a Bidirectional sequence-to-sequence model with fixed embeddings and attention. This paper talks about the writing style conversion focusing on Shakespearean literature. For the dataset, two sources have been used. Initially, for the collection of data on Shakespeare, different works of his have been used. Furthermore, over 4000 songs have been used for the rap lyric dataset and from the songs they got a total of 300,000 sentences. After that, the data was preprocessed to create a file of token ID and the rap lyric files transferred into token ID files. Moreover, after human evaluation and BLEU measurement metrics, the paper demonstrated that the Bidirectional sequence to sequence model with fixed embeddings and attention performed better than the previously used models, but the problem is: that it is changing the original meaning of the sentences. And so, future improvements include the fixing of such arising problems by incorporating other models with this to get more accurate outputs.

Another study authored by Zhijie Zhang, Zhongyuan Han, Leilei Kong, Xiaogang Miao, Zeyang Peng, Jieming Zeng, Haojie Cao, Jinxi Zhang, Ziwei Xiao and Xuemei Peng [1] researched about the detection of any style change of a text has occurred and also does the marking of that part based on the writing style. The data set of this specific paper is provided by the style change detection (PAN21 Authorship Analysis: Style Change Detection) [22]. In the dataset, 70% are training data (Number of texts, Number of authors, Number of paragraphs - 11,200, 17,051 77,252) [1], 15% are validation data (Number of texts, Number of authors, Number of paragraphs: 2,400 4,792 16,495) [1] and 15% are for testing (Number of texts: 2,400) [1]. The authors of the paper mainly focus on three tasks, which are finding whether there are many writers, finding the place in the text where multiple writers have been identified, and labeling of the texts. For the measurement of the similarity, they are using the BERT model. After that, the effectiveness of the paper was shown by giving an F1 score for the tasks.

The paper researched by Kalpesh Krishna, John Wieting and Mohit Iyyer introduced a new method called Style Transfer via Paraphrasing(STARP) for transferring the style of a text without changing its sentiments [12]. This mainly follows an unsupervised approach and the method has three parts to it. While evaluating this model with 23 related papers on style transfer, it has been seen that this model outperformed other SOTA models: state-of-the-art methods. For determining the performance metrics, some calculations were taken in accountability such as human error evaluation, transfer accuracy, semantic similarity, and fluency of the given input text for better and more accurate results. In this paper, the authors mainly worked with two datasets which include - Shakespeare's author imitation and the Formality transfer dataset. Shakespeare author imitation dataset has 37000 training data, and the formality transfer dataset contains 105,000 sentences. Furthermore, the author introduced a new dataset, which is called the corpus of diverse style (CDS).

Likewise, we have come to a paper authored by Shuohua Zhou [16], exploring the usage of deep learning for summarizing texts. Through this paper, we get to know that traditional natural language generation techniques primarily relied on templates, but deep learning technology breaks away from traditional reliance on templates, where it enables the autonomous learning of input-to-output mappings, leading to end-to-end solutions and a reduction in the need for human involvement. It tackles the challenge of extracting and using information from vast amounts of data. This paper introduces us to such a method that is very effective in the field of text generation. It is done by deep learning, to be more specific Recurrent Neural Networks (RNNs), Deep Neural Networks (DNNs) and Convolutional Neural Networks (CNNs). The most appropriate of them all is considered to be RNNs, which use powerful approaches to generate natural texts due to its potential in handling sequential data. The authors of this paper suggest such a data processing methodology that includes four phases for the handling of automatic summarization based on deep learning. Initially, it begins by text preprocessing, then semantic understanding of the original text is done to analyze its stylistic elements, the information reorganization using attention mechanisms is carried out, and finally, abstract generation is done by the utilization of RNNs.A summarization model is presented in this paper that

utilizes a hybrid neural network encoder and a decoder with an attention mechanism that combines the strengths of CNNs and RNNs. Using convolutional layers, the encoder can capture contextual interactions between target vocabulary units and nearby words, upgrading the quality of the original text representation.

In this paper the authors, Martina Toshevska and Sonja Gievska [21], delve into the realm of text style transfer methodologies employing deep learning techniques. Language, being influenced by cultural, individual, and social factors, requires the ability to adapt communication styles to different contexts. This paper explores the advancements enabled by deep neural networks, which have made text style transfer a vibrant field of research. The authors dive deeply into the groundbreaking innovations in neural networks with deep layers that have changed natural language processing and generation. This paper demonstrates text-style transfer's ability which facilitates users in many forms of linguistic complexities of several eras and civilizations. The art of deep learning gives a door into literary time travel from classic epics to modern style. Personal style, genre, formality, politeness, offensiveness, and sentiment are important mentions in the research. Studies have shown language variations depend on demographic groups which include distinctions in language usage between genders or different age groups. These findings might widely impact the human-computer interface and the virtual assistants that might not collide with the reader's preferred language. It also discusses specific writing styles, such as Shakespearean style, which has been explored for rewriting sentences for edutainment purposes. The research provides a list of publicly available datasets, categorized as parallel and non-parallel, each serving different purposes for style transfer tasks. Parallel datasets include the Shakespeare3 dataset, the GYAFC dataset, the caption dataset, and Cheng et al.'s email dataset, offering sentence pairs for a style transformation. Non-parallel datasets encompass Yelp, Gender, Amazon, SST, IMDB, Paper-News Titles, Gigaword, Political slant, Twitter, Reddit, Politeness, and Expertise datasets, which are used for various style transfer tasks such as sentiment, formality, politeness, and expertise. The author mentions the use of Amazon Mechanical Turk (AMT) for annotating and creating formal or informal versions of sentences. The evaluation of text style transfer involves preserving the semantic content of the original sentence and ensuring the quality of the transferred style. It aims to measure how well the original sentence's meaning is preserved and the quality of the generated style. Metrics like METEOR, BLEU, BERTScore, ROUGE-L, SARI, and PINC are used to evaluate content preservation. Pre-trained classifiers and precision/recall measures assess the quality of the generated style. Popular deep learning techniques for text generation are: Recurrent Neural Networks (RNNs), Long Short-Term Memory Networks (LSTMs), Gated Recurrent Units (GRUs), Convolutional Neural Networks (CNNs), and attention mechanisms are popular deep learning techniques for text generation, capturing sequential patterns and spatial features. Attention helps the model to focus on important sections of the input sequence. The literature review includes all the difficulties and goals of assessing text style transfer. The evaluation focuses on measuring both the keeping of content from the actual sentence and the standard of the style in the produced output. The review also presents various deep neural network designs widely used in text generation, such as Recurrent Neural Networks (RNNs), Long Short-Term Memory Networks (LSTMs), Gated Recurrent Units (GRUs), Convolutional Neural Networks (CNNs),

and attention mechanism. These methods have an essential function for displaying sequential data and preserving associations within the text. Various metrics are utilized for analyzing content preservation, including word overlap-based metrics like METEOR and BLEU, BERTScore, ROUGE-L, SARI, and PINC. These measures assess the similarity, accuracy, recall, and commonality between produced and reference texts.Assessing the efficacy of generating a particular fixed style, experts incorporate accuracy computations through pre-trained classifiers, along with precision, recall, and F1-measure. The attention mechanism involves self-attention and multi-head self-attention, which makes the network able to focus on the essential parts of the input sequence during the generation process. The document outlines several models in each and every category, that subsequently inform their methods and strategies. The models aid in creating output sentences that either recreate the input material or integrate style-specific properties using style classifiers. In the world of text generation, there is a pair of significant architectures. First, The encoder-decoder, and, second, Generative Adversarial Networks (GANs) respectively. The encoder is used to build a fixed-length vector that acts as an input sentence, whereas the decoder produces an output sentence utilizing the following representation. Autoencoders (AE) and Variational Autoencoders (VAE) are distinct systems, basically centered on gaining internal representations of input data. Generative Adversarial Networks (GANs), on the other hand, consist of a generator and a discriminator. These architectures offer powerful techniques for generating text and have proven effective in capturing and generating meaningful content. Some models incorporate a step that detects and removes style cues from input sentences. Style markers are phrases that significantly reflect a sentence's style. Various methods, such as n-gram salience measure, attention weights, importance scores, and monitoring style classifier probabilities, have been proposed to detect style markers. There are other two baseline models that use simplistic methods for sentiment modification. Additionally, the author explores deep learning models inspired by machine translation and paraphrasing. These models are categorized into three groups: simple reconstruction models, models with style classifiers, and adversarial models. The paper also mentions the use of additional components like back-translation, cycle reconstruction, private encoders, and decoders to improve style transfer. Finally, it highlights the importance of content preservation and fluency in generating meaningful and stylistically accurate sentences. The paper also discusses adversarial models for text style transfer. These models incorporate style discriminators similar to those used in GAN architecture. Various models within this category are described, along with their techniques and approaches. They aim to generate realistic and style-compatible sentences by using discriminators to determine the authenticity of the generated sentences. Some models also include additional classifiers to assist in the generation process. The paper highlights the collaborative feedback between the generator and the discriminators in these models.It motivates researchers to utilize deterministic and variational autoencoders and cycle-consistent constraints in order to unmask content and style in the latent space. To illustrate its view, the models include many evaluation factors—different loss functions and techniques to ensure content preservation and style transfer. Furthermore, here, the feature mover's distance and style discrepancy are presented as alternative loss functions in some models. The paper further explores its experiment using adversarial models for text-style transfer. These models are designed to generate sentences that not

only sound natural but also match a specific writing style. They do this by incorporating style discriminators, acting as "style police," to verify the authenticity of the generated sentences. Another thing is that the models use very insightful approaches including deterministic and variational autoencoders, to distinguish the content and style of the text. This separation allows the models to preserve their authenticity while transferring the desired style. The author's researcher utilizes deep learning models, namely Transformers, CAST, and StyIns, incorporating techniques such as adversarial style loss and multi-class discriminators. The Pre-train and Plug-in Variational Autoencoder (PPVAE) framework addresses the problem of learning new styles by using two variational autoencoders. One challenge is the availability and quality of datasets, as creating benchmark datasets covering diverse style categories is complex. Deep learning techniques are highlighted for their capabilities in automated feature learning and pattern recognition from large data, but improvements are needed to address overfitting and generalization issues. Future research priorities include style-content disentanglement, balancing content preservation and style strength, deep learning model interpretability, transfer learning, ethical considerations, and studying deep reinforcement learning. These developments have an impact not only on style transmission but also on domains such as linguistic creation, summarization, question-answering, and speech.

This paper by Bernhard Liebl & Manuel Burghardt [13] presents an approach for identifying Shakespearean intertextuality in contemporary fiction using computational methods. The authors of this paper have developed the Vectorian, a unique framework that utilizes different word embeddings and NLP parameters. The Vectorian functions like a powerful search engine that identifies similarities between sentences and where one can enter a Shakespearean phrase, and the system searches for passages in the books that likely contain that phrase, either directly or in a paraphrased form. It preprocesses the data by splitting it into sentences and assigning part of speech tags to tokens. Using contemporary word embeddings, it computes similarity scores between tokens. The system stores the data effectively and determines alignments based on these ratings. The engine includes customizable parameters, including options to omit determiners, apply semantic weighting, and penalize inconsistencies in part of speech tags. The authors conducted an ablation investigation to evaluate alternative configurations of embeddings and NLP settings, revealing insights for future studies in this area. Although the technique emphasizes evident references rather than nuanced ones, it fills a vacancy in computational methods to analyzing intertextuality and gives a significant instrument for understanding Shakespeare's effect in Shakespeare literature. In this work, the researchers evaluate several parameters utilized in the Vectorian search engine for computing embedding similarity and alignment score. These parameters include Embedding Interpolation (EMI) for combining different word embeddings, Embedding Similarity Measure (ESM) for determining the strategy to compute similarity scores, Inverse Frequency Scaling (IFS) to weight similarity scores based on token occurrence probabilities, Similarity Falloff (SIF) for rescaling scores, Similarity Threshold (SIT) to filter out low similarity scores, Mismatch Length Penalty (MLP) to penalize mismatched tokens, and Submatch Boosting (SBO) for assigning scores to partial matches. Their inquiry on similarity measurements, frequency scaling, similarity falloff, and thresholding enhances the accuracy of the search engine. Using a subset

of quotes from Shakespeare's Hamlet, they evaluated the system's performance using the Discounted Cumulative Gain metric. Using Optuna, the researchers adjusted the Vectorian search engine's parameters. For arithmetic mean (A) and harmonic mean (H) targets, the best configurations produced nDCGs of 77.6% and 75.2%, respectively. Some parameters had minimal impact, while others played a significant role. The embedding pipeline was not that useful for search results. Easy queries benefited from embeddings and syntactic markers, while hard queries did not. It highlights the importance of parameter customization based on query difficulty.

The paper authored by Md. Raisul Kibria & Mohammad Abu Yousuf [19], address the field of text generation in Natural Language Processing (NLP) and focuses specifically on Bengali language text generation. Although there have been numerous proposals for larger language models, the availability of context-driven text generation models, particularly for Bengali, is still limited. The authors suggest a bidirectional gated recurrent unit (GRU) based architecture for simulating the decoder of a sequence-to-sequence (seq2seq) model. Multiple context words are combined into a fixed-dimensional vector representation extracted from the GloVe language model. The Bengali Wikipedia dataset is used to train the baseline model, and beam search optimization is used to produce sentences with the highest cumulative log probability score. The model is compared against unidirectional LSTM and GRU networks using human scoring-based criteria. The experimental results show that the suggested model is effective at creating meaningful outputs that capture the goal context. The study advances the field of natural language processing (NLP) for the Bengali language and has applications in context-driven text generation. Like other papers, this research also shows an overview of various research works related to natural language generation and text generation models. Initially, Recurrent Neural Networks (RNNs) faced challenges due to vanishing or exploding gradient issues. To mitigate these problems, the multiplicative RNN (MRNN) model was proposed, combining RNNs with the Hessian-Free optimizer. LSTM addressed gradient issues and achieved better computational efficiency by incorporating gating mechanisms. Chinese poetry generation using RNNs involved the input of keywords and the use of multiple models to generate different lines of the poem. The encoder-decoder architecture, similar to the seq2seq model, became popular for text generation tasks. In the above cases, neural autoencoder models were used to construct paragraphs and documents while keeping syntax, semantics, and coherence intact. Another strategy, semantically controlled LSTM (SC-LSTM), was introduced for dialogue generation, integrating context vector values to control sentence planning. The paper explores various optimization techniques for language modeling. SGD is an iterative algorithm that calculates and applies gradient-based optimization for each training sample. RMSprop incorporates adaptive learning rates with mini batch gradient descent by keeping an exponentially weighted moving average of squared gradients. Adagrad is a customized version of SGD with adaptive learning rates for each parameter, suitable for online settings and sparse gradients. Adam uses momentum-based optimization in conjunction with RMSprop to achieve fast convergence and accurate approximation. The proposed framework is based on a generative conditional language model using word embeddings. Pre-trained GloVe embeddings are used, and different techniques are evaluated to combine context vectors. The neural network architecture includes separate networks for training

and inference. The paper overcomes the limitation of RNNs with arbitrary input lengths by using fixed-length sequences of 20. Left padding with zeros is applied to shorter sequences, yielding better results than right padding. Longer sequences are truncated from the beginning. Beam search is used during inference for better prediction by considering multiple probable outputs. Human evaluations are conducted to assess the generated text's quality, considering criteria such as sentence effectiveness, context representation, and relevance. Comparatively, LSTM architecture excels in learning syntax and semantics but struggles with preserving context, while the bidirectional GRU architecture achieves the highest overall score by effectively capturing context and formulating accurate sentences.

This paper authored by Ewoenam Kwaku Tokpo and Toon Calders [25], tackles the issue of bias in textual data, particularly in job advertisements and news publications, and introduces a text-style transfer model to mitigate bias automatically. The authors underline the negative impact of biased texts on target demographic groups, such as how masculine-sounding employment advertisements discourage female candidates. They highlight instances where automated language systems and AI tools have inadvertently learned and maintained biased behaviors, underscoring the need to address bias in textual data.The paper explores different approaches to style transfer, focusing on two main categories: auto-encoder sequence-to-sequence models and explicit style keyword replacement methods. It presents a text-style transfer approach that might automatically eliminate bias in the text through the combination of hidden information encoding with explicit keyword replacement. But according to the authors of the paper, since their model was continuously showing instances of low performance such as failing to withheld meaningfulness of a sentence and information loss, thus, they proposed a system where the methodology involves the successful transformation of a biased text to a neutral one all while preserving the semantic content of the text. Token Embedder, Token Decoder, Attribute Masker and Latent-content Encoder are the key components of this model, where the attribute masker helps to identify biased attribute words and replaces them with an uncommon symbol. Whereas, the token embedder's task is to produce embeddings for the masked tokens utilizing the BERT model that has been pre-trained on neutral texts. The Latent-content Encoder converts the original text into a latent-content representation, removing it from the influence of biased style. It employs dual-objective training to remove bias and ensure the generated representation is classified as neutral. In order to generate new token embeddings and forecast the correct tokens, the Token Decoder blends token embeddings and latent content. The model is evaluated on two datasets, demonstrating its effectiveness in style transfer accuracy, content preservation, and fluency compared to other models. In general, the proposed methodology strikes a balance between maintaining substance and minimizing bias in the text.

Apart from these, we have come across another interesting approach in a paper that was authored by Sheikh Abujar, Abu Kaisar Mohammad Masum, Md. Sanzidul Islam, Fahad Faisal and Syed Akhter Hossain where the technique of abstractive text summarization has been used to generate text in Bengali [8]. There are three major types of summarization methods- extractive, abstractive and keyword-based summarization. In extractive summarization, the important words or phrases are

taken from a given text, measuring their importance in the text and generating a summary. Similarly, in keyword summarization, only the keywords are taken to summarize the text which may not always make proper sense to us. However in the case of abstract summary, the machine tends to understand the structure of human written text to get an idea of the patterns between texts and phrases. This does not necessarily require to use of text from the input text for summarization, instead, it uses words and phrases from the corpus it was trained with to generate the texts that beholds the same meaning as the input text. Similarly, in the proposed model, the authors showed a process where the model to write a new sentence, used its previous learning patterns of human written sentences. [8] Here the concept of RNN has been elaborately analyzed and the author ultimately trained a model using LSTM, an advanced form of RNN to train their model of text generation. It is well known that RNN processes the sequence data incredibly well owing to its recurrent structure. The hidden units are updated at every epoch and don't have any limitation in sequence length and additional advantage that it provides is that both forward and backward computation aids the neurons in understanding the sequence. The authors made their own dataset adding necessary contractions and eventually building their own corpus. Finally, they were able to build a quite stable model that was able to generate Bengali text through abstractive summarization of a text with an accuracy of around 97%. However, they do have certain limitations, as the model cannot generate random length text [8]. The length had to be predefined. Also, to predict the next word, pad tokens had to be provided. Nevertheless, the authors have plans to overcome these limitations and build a better automatic text generator that will be able to provide a random-length Bengali text without using any predefined tokens or sequence phrases.

With this, we end our study on numerous scholarly research papers falling under the same category of work as ours because we successfully able to retrieve the necessary information to conduct our own research.

Chapter 4

DATASET

4.1 Collection of Dataset

In this part, we present a full overview of the procedures painstakingly implemented to assemble our large dataset, which comprises a wide variety of books produced by a broad range of Bengali authors. The data-collecting procedure evolved via a number of precisely planned processes, each aimed to capture the essence of Bengali literature.

Online Scraping for Novels: The data collection method commences with the selective exploration of online sources to build a proper dataset corpus. Through this procedure, we traversed the enormous expanse of the internet, methodically collecting a massive collection of literary masterpieces. As for our research purpose, the riches consisted of only novels credited to a diversity of Bengali literary heavyweights. The online scraping effort was not only massive, but it was also rigorous, ensuring that we captured a diverse range of genres, styles, and periods in Bengali literature. We ensured to keep a balance between the text datasets of Rabindranath Tagore and those of a few other authors.

PDF Acquisition: In addition to online scraping, we stretch our net further by collecting PDF editions of literary treasures. These PDFs are some crucial documents as they preserve the literary works in their original structure and formatting. By collecting the PDFs, we aim to retain the authenticity and integrity of the texts while permitting an entire investigation of these literary works.

OCR Techniques: The acquired PDFs, despite containing valuable content, posed the difficulty of being in a non-machine-readable format. To overcome this difficulty, we incorporated the precise use of Optical Character Recognition (OCR) methods. So, by utilizing OCR, we converted scanned pages into machine-readable text, ultimately transforming the PDFs into a digital repository of linguistic content.

Categorization of Data: Our dataset, reflecting the broad and varied Bengali literary environment, was deliberately divided into two basic sections for systematic study. The first category contained the works of Rabindranath Tagore, an iconic figure in Bengali literature whose impact stretches well beyond the confines of Bengal. The second group, as important, reflected the creative achievements of

Non-Rabindranath writers, including notable authors like as Sharat Chandra Chattapaddhay, Bankim Chandra Chattopadhyay Chattapaddhay, Kazi Nazrul Islam, Begum Rokeya, Humayun Ahmed, and Muhammed Zafar Iqbal. This category was crucial in organizing our later research, enabling us to dive deep into both the canonical and non-canonical components of Bengali literature.

Paraphrasing texts for building style transfer dataset: For carrying out the text generation task, we have paraphrased the sentence-level tokenized data of Rabindranath Tagore. Some data were paraphrased manually and some were done by using paraphrasing tools and we made sure to keep the data credible.

Existing dataset: We have collected some datasets from a former student and faculty of BRAC University named Asadullah al Galib who is currently working at BRAC IT. We have collected data of 3 authors from him and three are Rabindranath Tagore, Sarat Chandra Chattopadhyay and Bankim Chandra Chattopadhyay. We also cross-checked the validity of those datasets.

4.2 Data Preprocessing

Tokenization and File Organization: To allow an in-depth textual analysis, we utilized the proficiency of the BLTK tokenizer. This powerful program tactfully split the textual strings into individual tokenized sentences by separating each line according to their punctuation, which aided in a more organized and extensive analysis. Subsequently, these tokenized phrases were meticulously sorted into various csv files, each retaining the name of an individual author. This file arrangement not only helps data accessibility but also facilitates future author-specific analysis.

Data Purification: Recognizing the significance of data quality, we conducted a series of steps on the data with the intention of data purification. These codes were inclusive of diverse data cleaning methods, such as punctuation balance, correcting uncommon instances, removing incomplete sentences and guaranteeing data consistency by excluding too long sentences. Through these purification steps, our purpose was to obtain the apex of accuracy in data representation, creating a firm basis for our succeeding studies and discoveries.

These numerous data collection methodologies were carefully coordinated to create a dataset that not only depicts the breadth and depth of Bengali literature, but also upholds the highest standards of data integrity and relevancy. This comprehensive dataset serves as the foundation for our research, allowing us to delve into the deep nuances of Bengali literary traditions and their ongoing impact on culture and society.

4.3 Dataset Visualization:

1. **Sentence Length Analysis:**

   - **HUMAYUN AHMED:**

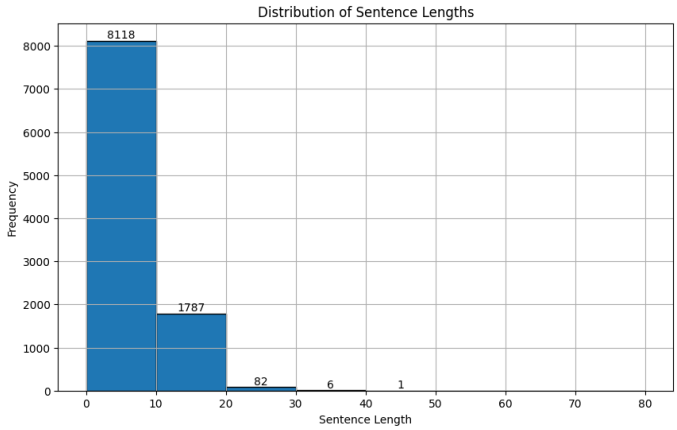


Figure 4.1: Humayun Ahmed Sentence Length

For Humayun Ahmed, the most frequent sentences are in between zero to 10 words, which is 8118 and the second most frequent is from the length of 10 to 20 words which comes to 1787 frequency and the least frequent is 20 to 30 words. 40 to 50 words is rare, which occurs only one time in our data set.

   - **RABINDRANATH TAGORE:**

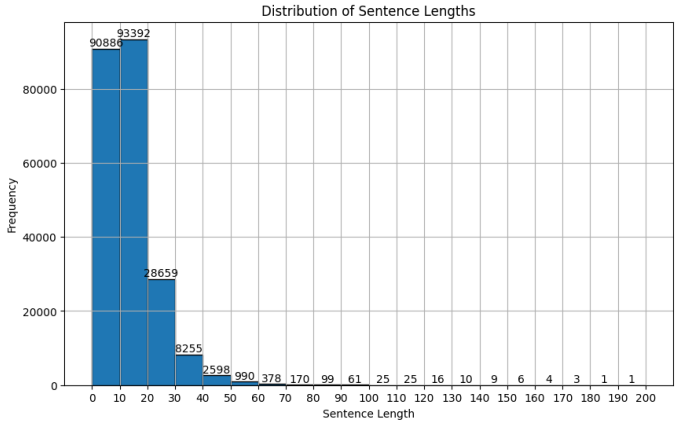


Figure 4.2: Rabindranath Tagore Sentence Length

For Ravindranath Tagore, the most frequent sentence length is 10 to 20 words, which occurs 93,392 times, and the second most frequent is zero to 10 words, which is 90,886. In our data set, we can see the least amount of frequency in the sentence length of 160 to 170 words. Also similar to

100 to 110 words.

- **SHARAT CHANDRA CHATTAPADDHAY:**

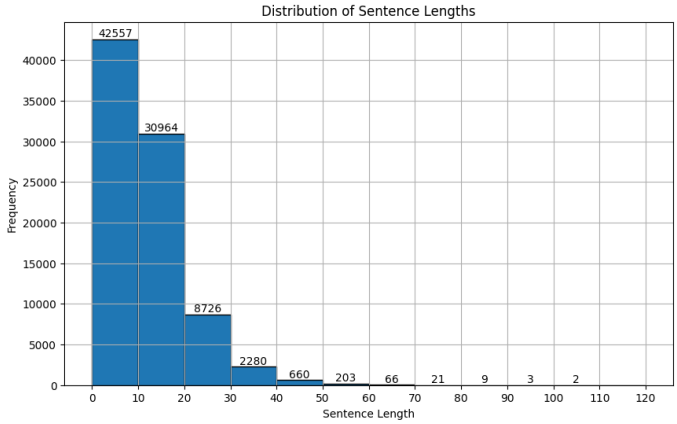


Figure 4.3: Sharat Chandra Chattapaddhay Sentence Length

For Sharat Chandra Chattapaddhay, the most frequent sentence length is from around zero to 10 words, which occurs 42,557 times in our data set. The next one comes at 10 to 20 words in sentence length which occurs 30,964 times and the least amount of sentence length is 40 to 50 words which occurs only 660 times.

- **Bankim Chandra Chattopadhyay:**

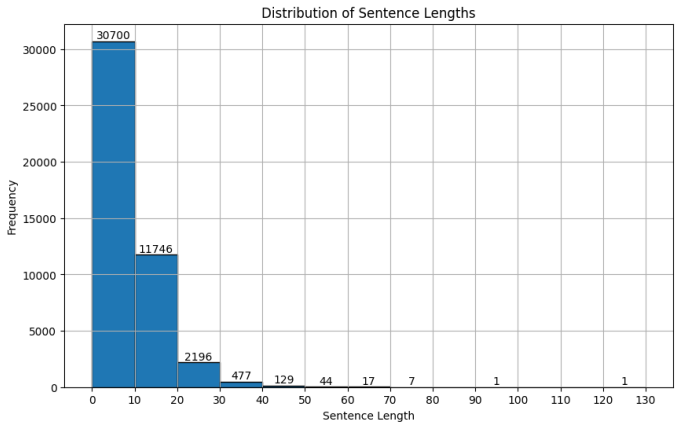


Figure 4.4: Bankim Chandra Chattopadhyay Sentence Length

For Bankim Chandra Chattopadhyay, he tends to write the most frequent sentence of zero to 10 words in the data set, it occurred 30,700 times. The second one is 10 to 20 words, which occurred 11,746 times, and the least amount of frequency in the sentence length is 60 to 70 words, which occurred 17 times.

- **ZAFOR IQBAL:**

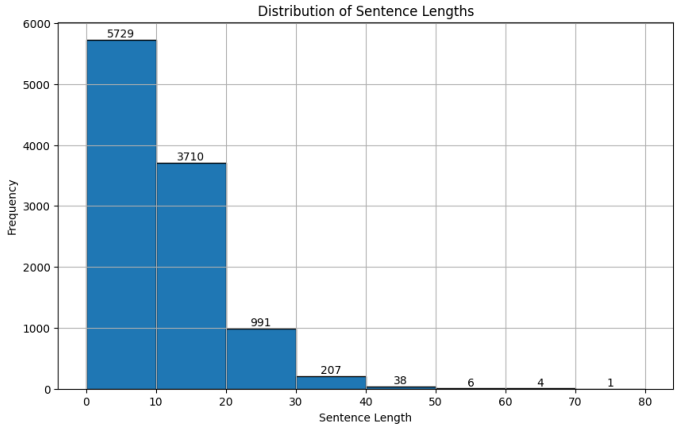


Figure 4.5: Zafor Iqbal Sentence Length

For Zafar Iqbal, the highest frequency of sentence length is zero to 10 words, which occurs 5729 times, and the least amount of sentence length is 40 to 50 words, which occurs 38 times. And the rare occasion is 70 to 80 words, which occurs only once.

- **BEGUM ROKEYA:**

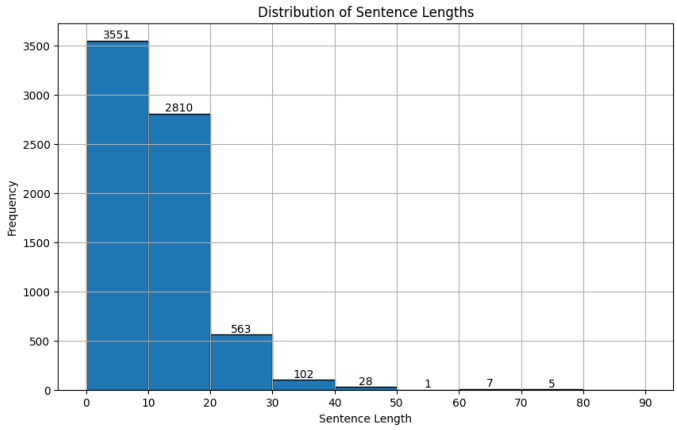


Figure 4.6: Begum Rokeya Sentence Length

For Begum Rokeya, she tends to write the highest frequency of sentences from zero to 10 words, which occurs 3551 times and the least amount is 40 to 50 words per sentence. These occurred only 28 times and the rare occasion is 50 to 60 words, which only occurred once in our data set.

- **AHMED SOFA:**

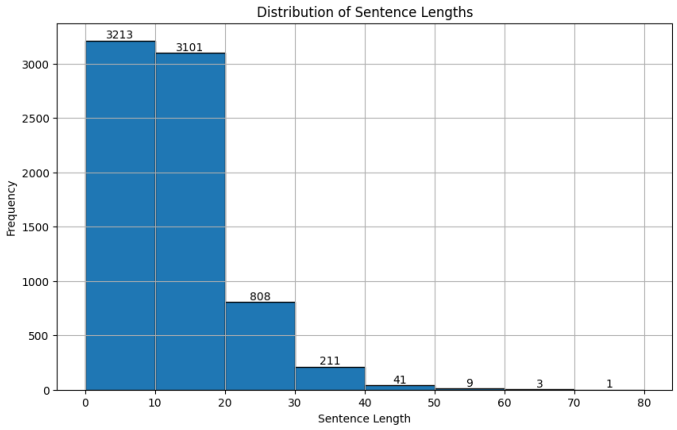


Figure 4.7: Ahmed Sofa Sentence Length

For Ahmed Sofa, He likes to write zero to 10 words in a sentence, and that frequency is 3213 times. The second amount is 10 to 20 words per sentence, which occurs 3101 times in our data set, and the least amount of words per sentence he likes to write is 60 to 70 times, which occurs three times in our dataset. Only one occurrence where he wrote 70 to 80 words in a sentence.

2. **Text Exploration Analysis:**

- **Bankim Chandra Chattopadhyay:**
  The graph image shows the word frequency analysis of a dataset of texts written by Bankim Chandra Chattopadhyay. The x-axis denotes the words, while the y-axis represents the frequency of each word, with the most frequent words positioned at the graph's top. Notably, the term করে stands out with a frequency nearing 8000, whereas the word হয় has the lowest frequency, falling below 200. It is noteworthy that the most frequently used word surpasses 4000 occurrences, setting it apart. Furthermore, the graph displays a gradual decrease from the second bar, steadily covering words from "বলল" to "তার", followed by a relatively stable trend from "থেকে" to "হয়". The average frequencies of words like "দিয়ে", "কি", "থেকে", আর, "করতে", "কথা", "নিয়ে", "সে", "না", "তখন" etc., are relatively uniform, around 2000.

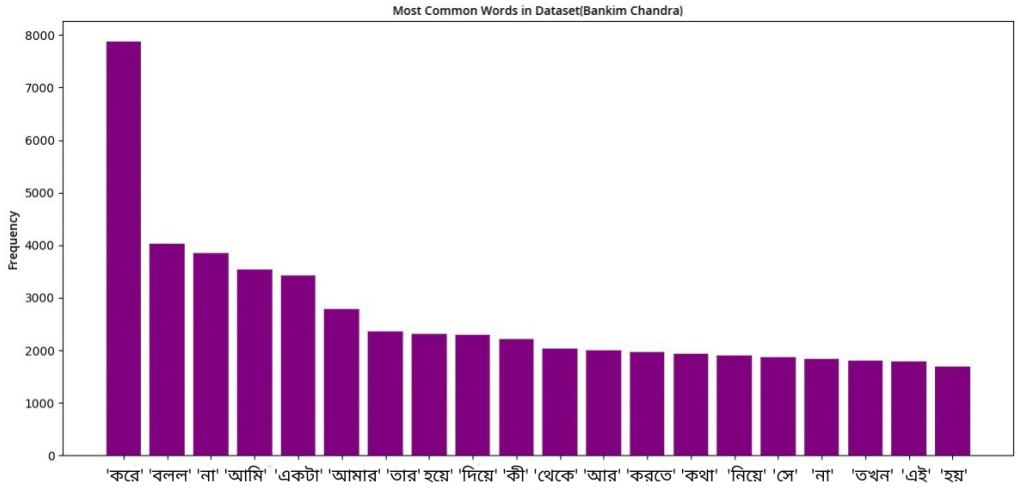


Figure 4.8: Bankim Chandra Chattopadhyay Word Frequency

- **BEGUM ROKEYA:**
  The graph image showcases the word frequency analysis of a dataset of texts written by Begum Rokeya. There, the x-axis shows the words, and the y-axis shows the frequency of each word. Words that occur most frequently appear prominently at the peak of the graph. Upon initial inspection, the graph appears consistently steady. The analysis highlights that the word "না" holds the highest frequency in Begum Rokeya's writings. In contrast, the word "আমার" appears with the lowest frequency, approximately 318 times. An observation of the entire graph reveals a steady frequency range for words regularly utilized by the author, spanning from 300 to 400. Additionally, the initial section of the graph depicts a gradual decline among words such as "যে", "হয়", "আমি", "করিয়া", and "আর" starting at around 600 and descending to approximately 450.

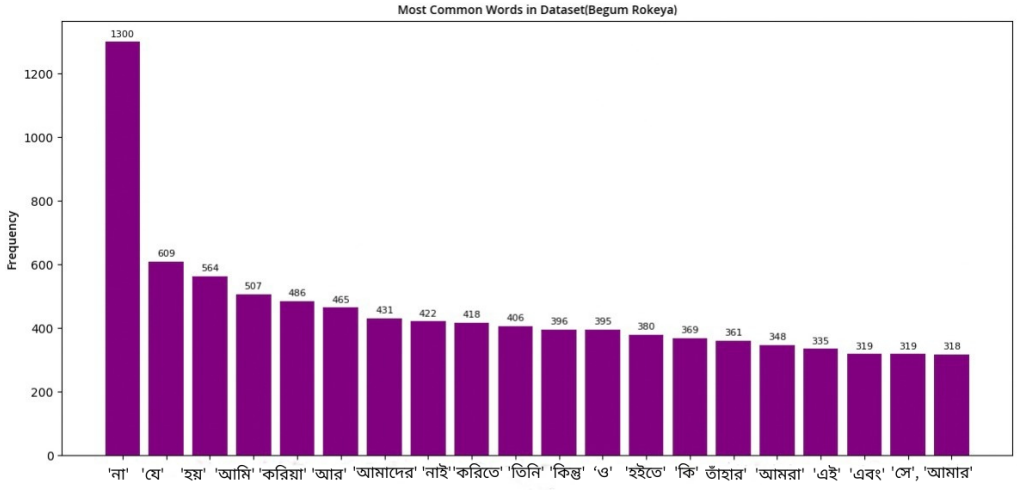


Figure 4.9: Begum Rokeya Word Frequency

- **HUMAYUN AHMED:**
  In the graph, the x-axis shows the words, and the y-axis shows the frequency of each word, where the graph illustrates word count analysis from a data collection of texts by Humayun Ahmed. Words with the most recurrence are seen at the uppermost section. In this segment, it is evident that the most frequently used word by the author, Humayun Ahmed is "না" occurring 1627 times in our dataset, which markedly distinguishes it from other words he used. On the other hand, the least used word is কিছু with a frequency rate of 263. While examining the data, it becomes evident that the words he used the least have relatively similar frequencies, such as "বলল", এক, "মনে", "বললাম", "কিছু". Additionally, there is a noticeable similarity in frequency "হয়", "সে", "সঙ্গে", "হয়ে", "বলল", all ranging approximately between 300 and 350 occurrences.

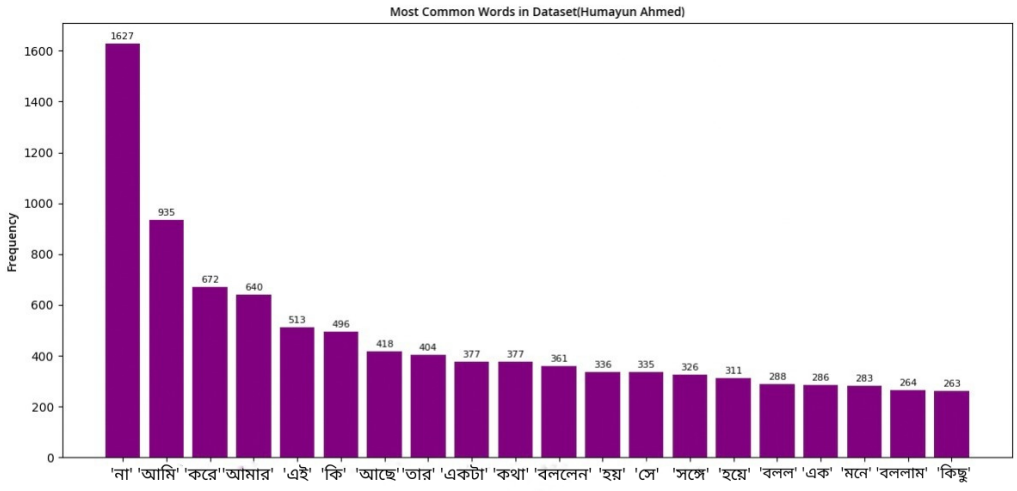


Figure 4.10: Humayun Ahmed Word Frequency

- **RABINDRANATH TAGORE:**

This chart presents an analysis of the frequency of words in a collection of texts authored by Rabindranath Tagore. On the horizontal axis, we can observe the words, while on the vertical axis, it represents the frequency of each word. Words with the highest occurrence are positioned at the graph's uppermost section. The globally renowned author Rabindranath Tagore employed the word "না" the most, utilizing it a remarkable 48,647 times. In contrast, the least used word is "মনে" occurring only 11,692 times. Furthermore, an observation indicates a consistent frequency among words such as "কিন্তু", "আমাদের", "তার", "আমি", "সেই", "তাহার", "মধ্যে", "হয়" with occurrences ranging from 15,000 to 17,000. There is also a consistent decline from "সে" to "কিন্তু".

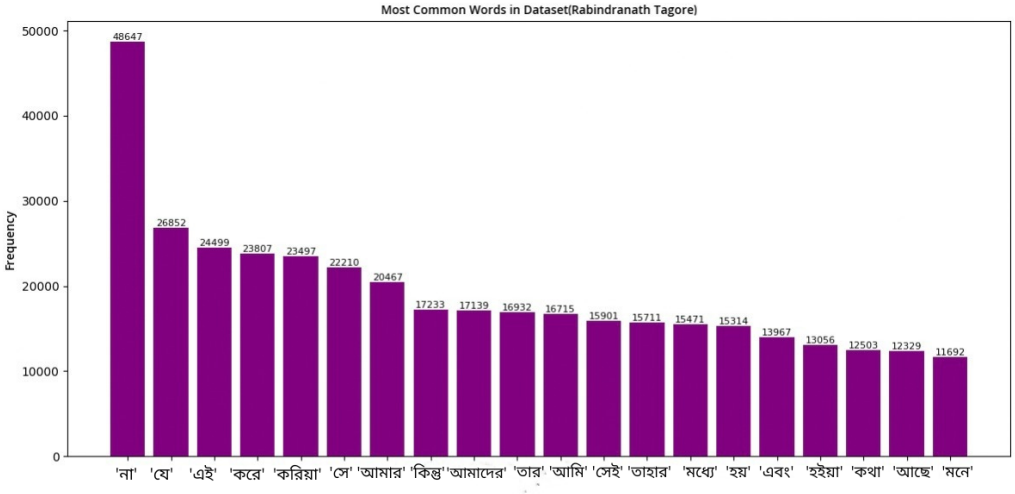


Figure 4.11: Rabindranath Tagore Word Frequency

- **SHARAT CHANDRA:**

  The graph illustrates word count analysis from a data collection of texts by Sharat Chandra Chattopadhyay. The x-axis represents the words, whereas, the y-axis indicates their occurrences of them. Words with the highest occurrence are positioned at the graph's uppermost section. In this graph, a prominently utilized word by Sharat Chandra Chattopadhyay is "না", which stands out distinctly with a frequency of 25,320. Beyond this, there is a general fluctuation in the word frequencies throughout his writing. For instance, he has employed words like "সে", "করিয়া", "কিন্তু", "কি" at a frequency of around 10,000, whereas words like "করে", "কথা", "এ", "মনে", "একটা" have a frequency closer to 5,000. Additionally, a consistent pattern is observed in the graph with words like "আর", "হইয়া", "এই", "কহিল", "আমি" maintaining an average frequency of approximately 7,000.

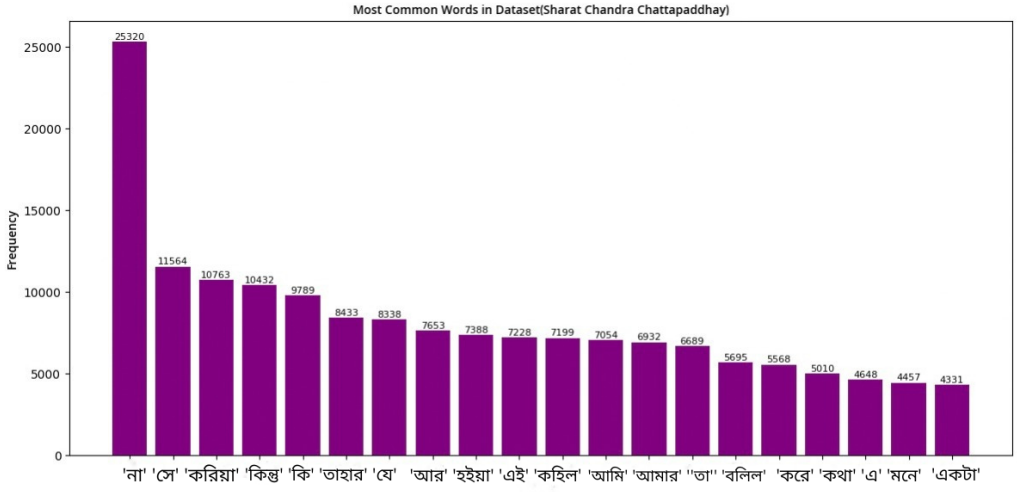


Figure 4.12: Sharat Chandra Word Frequency

- **ZAFOR IQBAL:**

  The chart shows the word count analysis from a collection of texts by Zafor Iqbal. The x-axis represents the words, while the y-axis indicates their frequencies. Words with the most appearances are prominently displayed at the graph's top. In contrast to the previously discussed authors, Zafor Iqbal stands out for his frequent use of the word "করে" in his writing. This particular word appears with a frequency of 2,247, and the second most frequently used word is "বলল" which is nearly as prominent. It is noteworthy that there is a distinct and significant decrease in frequency from "বলল" to "একটা" with a gap of around 1000. Additionally, his writings reveal a unique pattern, where the least used words are "এই" and "হয়" with consecutive occurrences of 515 and 507. The average frequency of words employed by Zafor Iqbal, such as "আর", "করতে" and "কথা" is approximately 600.

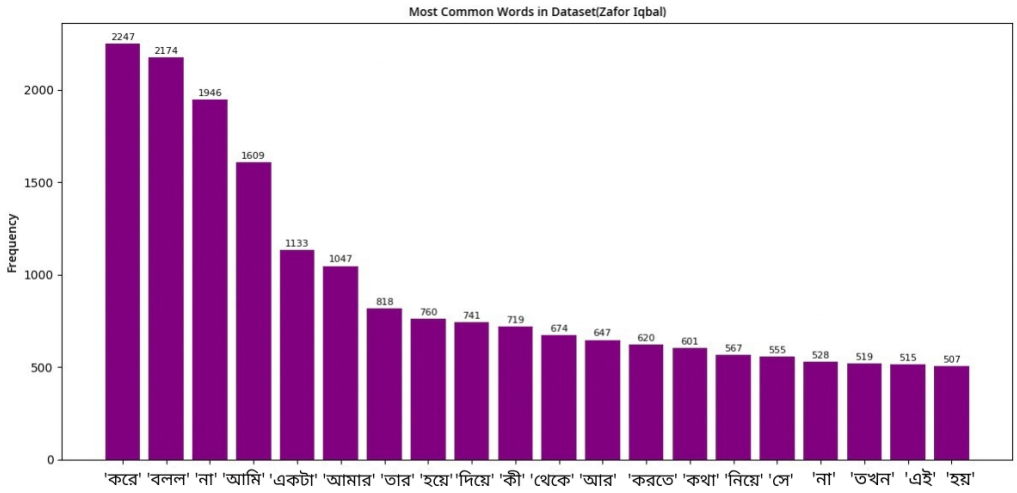


Figure 4.13: Zafor Iqbal Word Frequency

- **Ahmed Sofa:**

The graph displays the word frequency analysis of a dataset of texts written by Ahmed Sofa. The x-axis is for the words, and the y-axis shows the frequency of each word. The words that have the highest frequency are at the highest level of the graph. Analyzing the dataset of Ahmed Sofa, it's apparent that he frequently employs words like "করে","আমি", "না" and "তার" where "করে" being the most prominently utilized. Besides "করে", other frequently used words fall within a similar frequency range. However, there is a notable and sudden decrease in frequency from "আমার" to "করতে", dropping from 1122 to 599. Words like "করতে", "সে" and "একটা" exhibit similar frequencies, closely approaching 600. Notably, the least used words in his writing are "তা" and "এ". An overall fluctuation is observed in the graph, ascending and descending throughout the analysis.

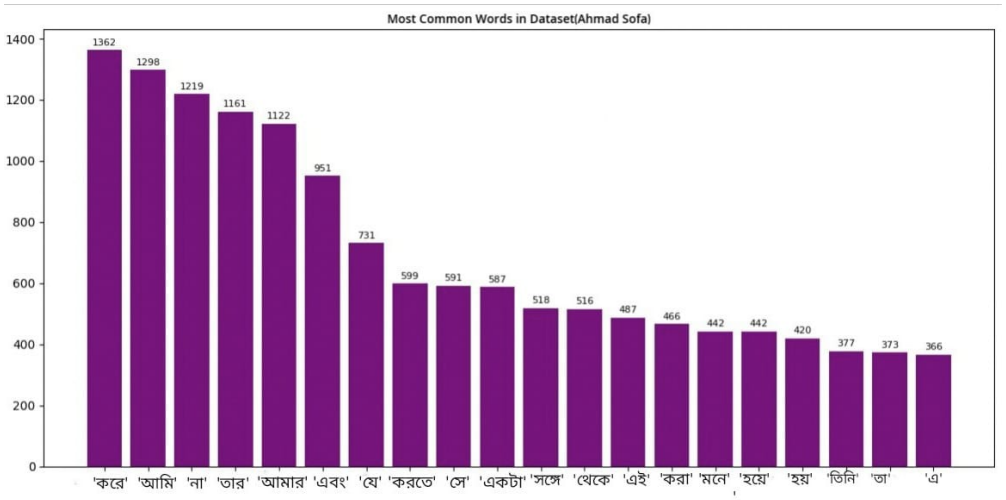


Figure 4.14: Ahmed Sofa Word Frequency

3. **Combined Data Analysis:**

- **Count of texts of different lengths:**
  For the combined results we get a large amount of 1,84,745 sentences. And for the medium we get the number to 1,24,556. Finally, for the small length sentences, the number is 82,735.

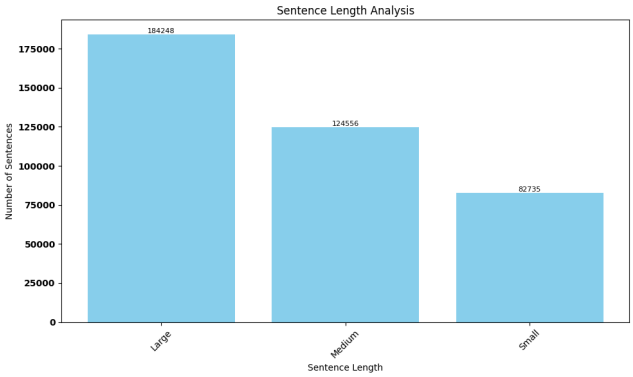


Figure 4.15: Sentence length Count

- **Text count of different Authors:**
  For the text count of different authors, the highest count belongs to Rabindranath Tagore with the number 2 ,25,593. On the other hand, the lowest number of sentences are from Begum Rokeya with 7067 sentences and in between Sharatchandra and Bankimchandra respectively have 85,491 and 45,321 sentences.

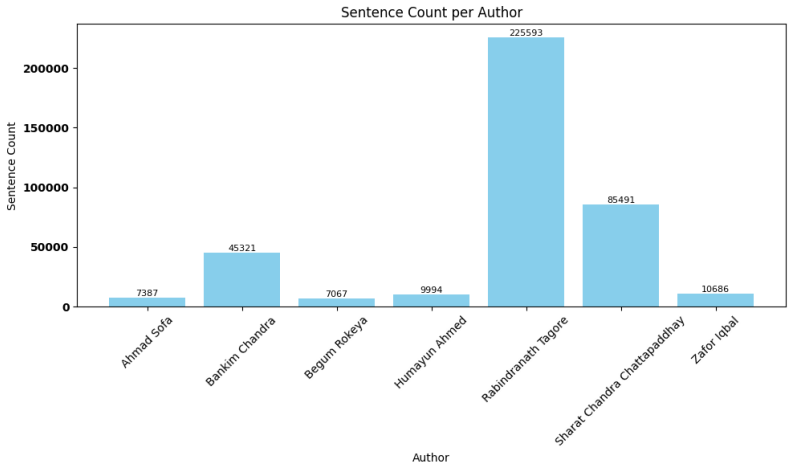


Figure 4.16: Author Text Count

Chapter 5

MODEL DESCRIPTION

5.1 Models used for classification

In this section, we present a comprehensive overview of the models designed and implemented to address the diverse classification tasks within the scope of our research paper. Our primary focus revolves around four distinct challenges: discriminating between Rabindranath Tagore's literary works and non-Rabindranath texts, distinguishing texts from the Rabindranath era and the contemporary era, classifying texts based on multiple authors, and identifying potential adversarial attacks. The development of robust models for these tasks involves a careful combination of natural language processing techniques, feature engineering, and advanced deep learning architectures. In the following subsections, we delve into the models used in the classification tasks, elucidating the architecture, training strategies, and evaluation methodologies employed to achieve meaningful results. In this section, we hereby explain in detail the methodologies we have incorporated with our model to correctly complete the task of author identification and text generation.

5.1.1 Models

Bidirectional LSTM Model:

A good model for extensive natural language processing (NLP) tasks such as sentence author identification would utilize Recurrent Neural Network (RNN). This is because it is quite able to capture sequential associations and maintain contextual information in a text data. RNN has received considerable attention because of its enhanced capacity to preserve sequence information as time passes [3]. On the other hand, a Bidirectional Long Short-Term Memory (Bidirectional LSTM) is a version of the Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) that evaluates input sequences in both forward and reverse directions, moreover, this is an advantage for this type of model because it is able to collect knowledge from both its past and future contexts which helps it understand the input text better. By effectively boosting the amount of information accessible to the network, BiLSTMs enhance the context extraction that the algorithm has access to (for example, by letting the algorithm know what words immediately follow and precede a word in a phrase).
To understand the concept more precisely we can show a breakdown of the components and operation of a Bidirectional LSTM:

1. **Long Short-Term Memory (LSTM):** LSTM is a form of recurrent neural network (RNN) that is meant to address the vanishing and exploding gradient issues associated with standard RNNs. It maintains a cell state to store and transmit information over several time steps, enabling it to retain long-term dependencies in the incoming data. The LSTM network design comprises of a few sections- forget gate, input gate, update gate and the output gate, and each portion performs an independent purpose.

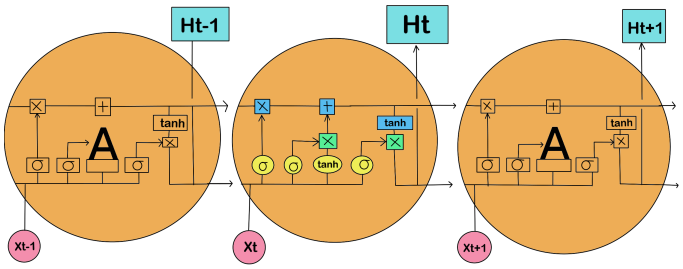


Figure 5.1: LSTM

- **Forget Gate:** In a cell of the LSTM neural network, the initial step is to determine whether we should maintain the information from the previous time step or forget it. It is done in the forget gate. The forget gate determines what information to discard from the cell state from the previous time step.

$$f_t = \sigma(W_{if} \cdot x_t + b_{if} + W_{hf} \cdot h_{(t-1)} + b_{hf}) \tag{5.1}$$

x_t : Input at time t
$h_{(t-1)}$: $Hidden state of the previous time step.$
W_{if} and W_{hf} : Weight Matrices for the input and hidden state, respectively.
b_{if} and b_{hf} : Bias vector for the input and hidden state, respectviely.
σ : Sigmoid active function.

- **Input Gate:** The input gate controls how much information should be stored in the cell state from the current time step.

$$i_t = \sigma(W_{ii} \cdot x_t + b_{ii} + W_{hi} \cdot h_{(t-1)} + b_{hi}) \tag{5.2}$$

$$\tilde{C}_t = \tanh(W_{ig} \cdot x_t + b_{ig} + W_{hg} \cdot h_{t-1} + b_{hg}) \tag{5.3}$$

x_t : Input at time t
$h_{(t-1)}$: $Hiddenstateofthepreviousstimestep.$
$W_{ig}\ and\ W_{hg}$: Weight Matrices for the input and hidden state, respectively.
$b_{ig}\ and\ b_{hg}$: Bias vector for the input and hidden state, respectively.
$\tilde{C}_t$: Hyperbolic tangent activation function.

- **Update Gate:** The candidate cell state is the new information to be added to the cell state, computed using the input gate.

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{5.4}$$

$\odot$: Element-wise multiplication

- **Output Gate:** The output gate controls how much information from the cell state is used to compute the output at the current time step.

$$o_t = \sigma(W_{io} x_t + b_{io} + W_{ho} h_{t-1} + b_{ho}) \tag{5.5}$$

$$h_t = o_t \odot \tanh(C_t) \tag{5.6}$$

$W_{io}\ and\ W_{ho}$: Weight Matrices for the input and hidden state, respectively.
$b_{io}\ and\ b_{ho}$: Bias vector for the input and hidden state, respectively.
C_t : Cell state at time t.

2. **Architecture:** The architecture of a Bidirectional LSTM typically consists of two LSTM layers: one processing the input sequence in the forward direction and the other in the backward direction. Each LSTM layer has its own set of parameters, including weights and biases.

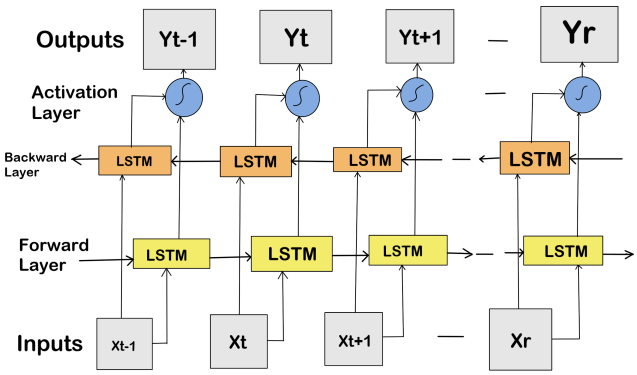


Figure 5.2: Bi-directional LSTM

3. **Bidirectional Processing:** In a Bidirectional LSTM, the input sequence is processed in two directions: forward and backward. The forward LSTM processes the sequence from the beginning to the finish, whereas the backward LSTM processes it in reverse. This dual processing allows the model to learn dependencies from both past and future contexts.

4. **Input and Output:**

   - **Input:** Each time step of the input sequence is supplied as input to both the forward and backward LSTMs.
   - **Output:** For each time step, the outputs from the forward and backward LSTMs are concatenated, essentially doubling the number of features at each time step. This concatenated output is used as the input to the following layers or for generating predictions.

5. **Training:** During training, the parameters (weights and biases) of both the forward and backward LSTMs are changed via backpropagation through time (BPTT). The loss is calculated based on the model's predictions and the ground truth labels.

Given the features and context-extracting characteristics of Bidirectional LSTM, we selected it for our text author classification.

BanglaBERT:

BanglaBERT uses the Transformer architecture, recognized for its effectiveness in Natural Language Processing (NLP) activities. This design depends on encoders and decoders with self-attention mechanisms to understand links between words in a phrase and is highly efficient. BanglaBERT is particularly built for the Bengali language. Its vocabulary and training data are tuned to interpret and create Bengali text. BanglaBERT is pre-trained on a massive 27.5 GB dataset (called 'Bangla2B+') by crawling 110 notable Bangla sites, guaranteeing it learns the precise qualities and patterns of the language [17]. This pre-trained information may be utilized to numerous downstream NLP tasks. The pre-trained BanglaBERT may be fine-tuned for particular tasks like sentiment analysis, text summarization, machine translation, or question answering. This is known as transfer learning, when the model's previous knowledge is applied to a new domain. By evaluating semantic relationships and extracting relevant features from Bengali text, BanglaBERT may be used for tasks like named entity identification, topic modeling, or relationship extraction. BanglaBERT also includes modifications for text production reasons and operates by gaining an extra decoder component. This decoder, often based on a Transformer design as well, takes the encoded representation from BERT and gradually makes text tokens one by one. Then, for specific text generation tasks, the model (both encoder and decoder) is fine-tuned on relevant datasets to tailor its output to the desired task [23].

Architecture Overview of BanglaBERT:

- **Transformer-based Encoder:**
  - Based on the Transformer design, noted for its intense attention mechanism.
  - 12 encoder layers, each having self-attention and feed-forward networks.
  - Processes the input Bengali text, capturing associations between words and constructing contextual representations.
- **Decoder (for text generation tasks):**
  - Often added to the base BanglaBERT for producing additional text sequences.
  - Takes the encoded representation from the encoder as input.
  - Sequentially creates words one by one, each conditioned on the preceding ones.
  - May utilize beam search or top-k sampling for diverse and fluent generation.

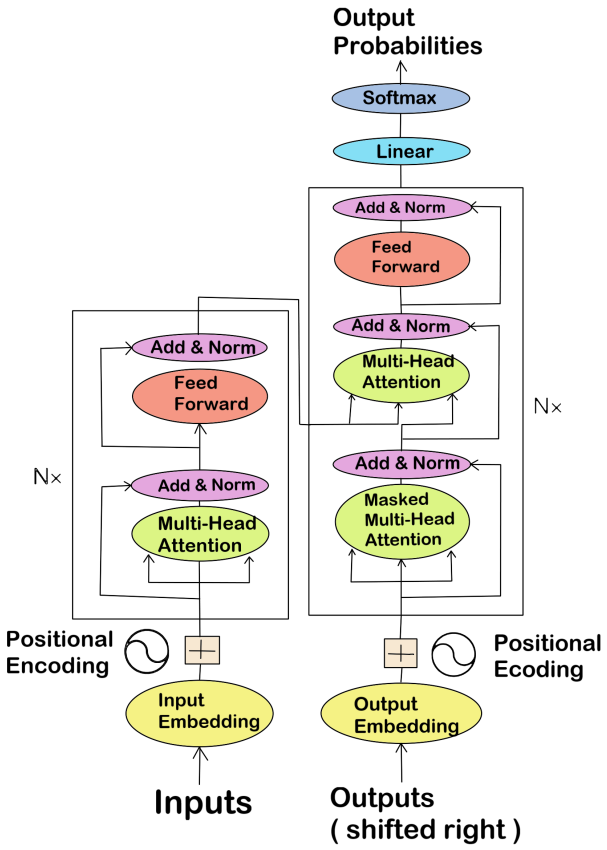


Figure 5.3: Transformer Model

5.1.2 Experimental Setup of the used models

i. Experimental Setup for Rabindranath vs. Non Rabindranath Texts:

Duplicate Sentences Removal:

In our trial arrangement, the main purpose was to detect and delete duplicate terms from the data collection. This approach was used to reinforce the data's legitimacy and remove any possible biases produced by repeated words. By calculating the overall number of phrases and then determining the number of unique ones, we were able to uncover instances of duplication. These repetitive words were later deleted, ensuring that each sentence in the data collection offered separate information for the model's learning and assessment stages. The adoption of this basic strategy coincides with conventional processes in data pretreatment for tasks linked to natural language processing, hence boosting the quality and dependability of our experimental setup.

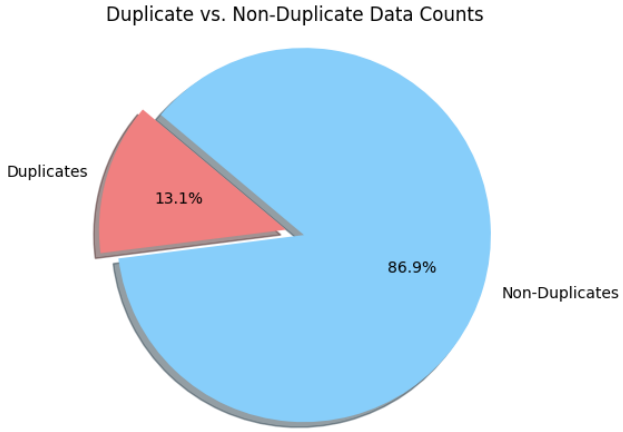


Figure 5.4: Duplicate Amount

Author Labeling:

To optimize our categorization effort, we gave a category label to each author in the dataset. This was accomplished by adding a `'Author_label'` column to the DataFrame. The labels were allocated numerically using a mapping. `'Rabindranath Tagore'` was given a value of 1, while all other writers, such as `'Bankim Chandra'`, `'Sharat Chandra Chattapaddhay'`, `'Humayun Ahmed'`, `'Zafor Iqbal'`, `'Begum Rokeya'`, and `'Ahmad Sofa'` were awarded a label of 0. The act of categorizing in this manner simplifies the following process of classifying, allowing the model to efficiently learn and distinguish between authors during the training phase.

Undersampling:

1. **Train Data:** During the undersampling step, we mitigated class imbalance by generating balanced subsets for each category of authors. Specifically, 109464 instances for category 1, 61188 for category 2, 28113 for category 3, 4438 for category 4, 5320 for category 5, 4944 for category 6, and 5459 for category 7 were randomly picked or replaced. The final train_data DataFrame combines

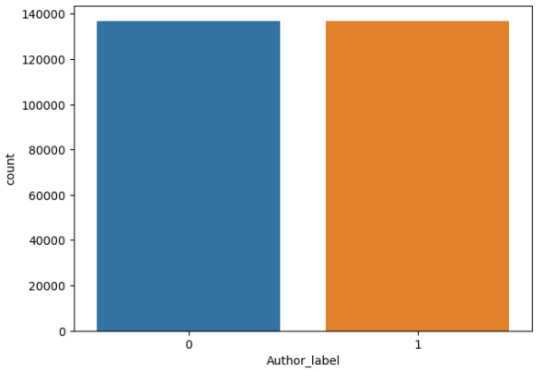


Figure 5.5: Train Data

both subsets, guaranteeing enough representation for each author category and increasing the model's generalization during training and assessment.

2. **Test Data:**

   In the undersampling phase of our experimental design, the major goal was to address the uneven distribution of authors within the dataset. The technique entailed constructing unique subgroups for each author type, modifying the size of these subsets to remedy the imbalance. For several categories, random selection with replacement was applied to boost the presence of minority writers. These subsets were then pooled to produce both the training and assessment datasets, guaranteeing a more fair distribution of cases across various author categories.

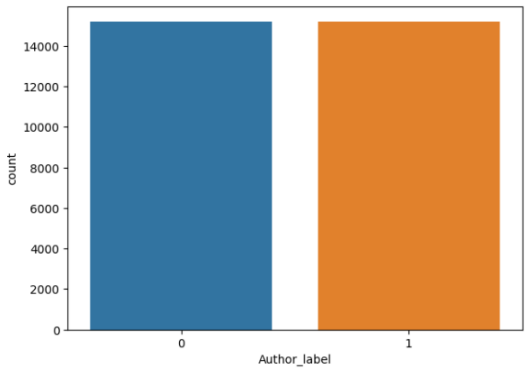


Figure 5.6: Test Data

Subsequently, a similar strategy was utilized to build the test dataset. Specifically, 15204 instances from df11, 6622 instances from df12, 4786 instances from df13, 1046 instances from df14, 1005 instances from df15, 913 instances from df16, and 832 instances from df17 were utilized to produce the test data subsets. This ensures a balanced representation for a fair assessment of the model's performance. The shuffling of cases across these datasets sought to create unpredictability, further boosting the model's capacity to generalize successfully across distinct authors. This careful undersampling method created the groundwork for a more robust and fair assessment of our model.

Model Configuration (BERT + Bidirectional LSTM):

Incorporating BanglaBERT for Textual Embeddings: Within our experimental framework, we implement BanglaBERT, a transformer model particularly developed to solve problems in the Bengali language. Leveraging the features of the Hugging Face Transformers library, we add the BanglaBERT model and tokenizer for contextual text embeddings. Furthermore, the incorporation of the 'normalizer' module serves a role in fine-tuning the text document. These selections align with the latest advances in Natural Language Processing (NLP), ensuring that our model effectively processes Bengali text. The integration of BanglaBERT strengthens our experimental context and opens up possibilities for successful implementation in various NLP applications.

Data Preparation for BanglaBERT Integration: The process of preparing input data is a comprehensive one, covering both phases—training and testing. We use a BanglaBERT tokenizer to turn sentences into numbers (input IDs), and we create attention masks, crafted to emphasize the significant parts(tokens). The dataset is then divided distinctively into training and testing subsets, making arrays with input data (X_train_input and X_test_input), associated labels (Y_train_label and Y_test_label), and attention masks (train_mask and test_mask). An important train-test split is applied to these arrays for training and testing. Thereby, the procedures facilitate the effective integration of BanglaBERT into our experimental context, creating a pathway for the successful execution of natural language processing tasks.

Applying Padding to Sequences: In the framework of Bidirectional Long short-term Memory (LSTM), the process of padding sequences incorporates adding specific tokens or elements to input sequences to manage a uniform length. This step is essential for our training, validation, and test texts as LSTMs are sort of designed with fixed-size inputs, whereas text sequences in natural language processing (NLP) applications typically vary in length. The implementation of the pad_sequences function from the pad_sequences package is utilized in padding the sequences for training neural networks. Once the padding sequences are established, we proceed to construct our Bidirectional Model.

Model Architecture:

In this part, we train a Bidirectional LSTM model for the goal of author classification, associating with the BanglaBERT embeddings. The model architecture is outlined as follows:

- **Input Layer:** The model accepts input sequences with a maximum length of `max_len` tokens.
- **BanglaBERT Layer:** Leveraging the BanglaBERT model, input sequences and attention masks are supplied to record contextual embeddings.
- **Bidirectional LSTM Layer:** A Bidirectional Long Short-Term Memory (LSTM) layer is presented, containing 64 units, a dropout rate of 20

- **Output Layer:** A dense layer with a softmax activation function generates probabilities for binary classification, expressing the probability of each author. The model seeks to categorize input sequences into one of the two author groups.

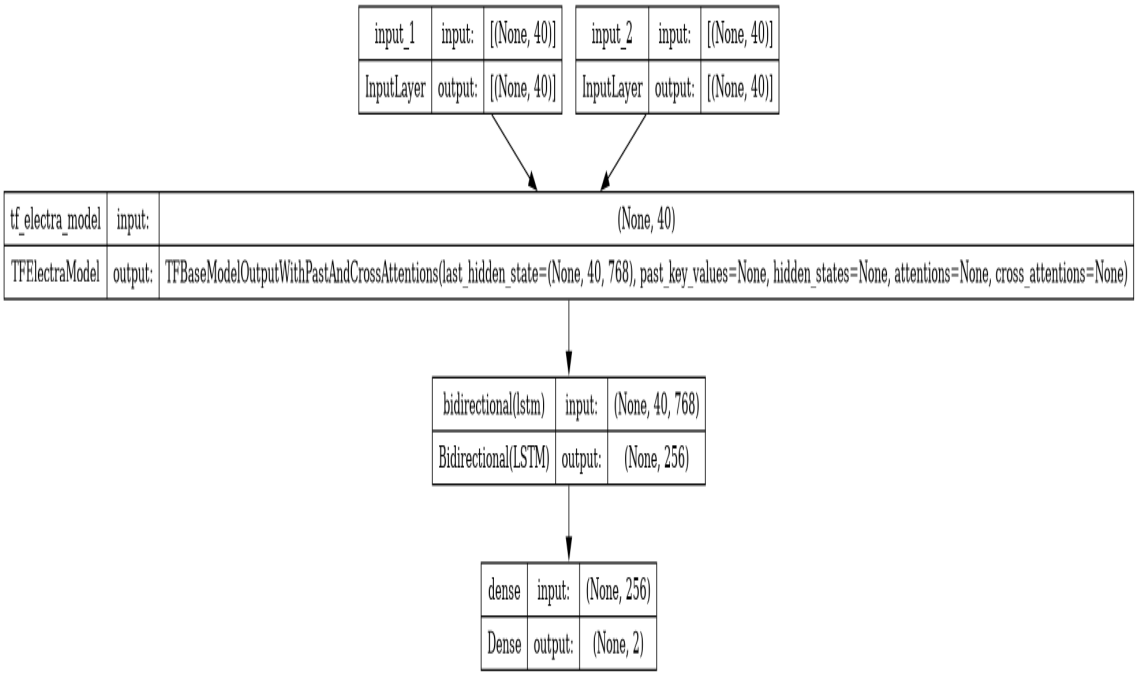


Figure 5.7: Model Plot

Model Setup:

We have set up a model in the setup and compilation process, with key markers that drive the training process. Here's an explanation of the metrics and other parameters:

- **Loss Function:**
  Sparse Categorical Crossentropy: This is the selected loss function for multi-class classification jobs, which includes allocating each instance to a single class. It calculates the cross-entropy between actual labels and predicted probabilities, allowing effective training by penalizing those who stray from what reflects the true class.

- **Metrics:**
  Sparse Categorical Accuracy: This statistic evaluates the accuracy of the model's predictions. It is appropriate for multi-class categorization and provides insights into the percentage of accurately classified occurrences.

- **Optimizer:**
  Adam Optimizer: This is an optimization strategy that modifies the learning rates of each parameter separately. It is a typical approach for deep networks, providing rapid convergence and adaptability to diverse learning rates.

Model Fitting:

1. **Batch Size:**
   During training, the data is split into batches, each having 32 samples (`batch_size=32`). Employing this batch-wise processing assists in optimizing memory use and speeds up the training process.

2. **Epochs:**
   The model undergoes several iterations throughout the full training dataset, referred to as epochs. In this case, the training technique is iterated for 500 epochs (`epochs=500`), enabling the model to steadily increase its performance over time.

3. **Early Stopping:**
   The EarlyStopping callback is added to cease training if there is no substantial improvement in the validation loss after one epoch (`patience=3`). This preventative strategy mitigates overfitting and guarantees the model generalizes efficiently to fresh inputs. The `min_delta` parameter determines the smallest change in the monitored measure necessary to qualify as an improvement.

Model Configuration (Bidirectional LSTM):

Train-Evaluation Split:

- **Purpose:** Preparing the combined dataset for subsequent splitting.
- **Operation:** Utilizing the `train_test_split` function to allocate 80% of the data for training and 20% for evaluation.

Textual Data Preparation for Model Input:
In this phase of our experiment, we employ tokenization and padding techniques to prepare the textual data for effective model input.

Tokenizer Configuration:

- **Purpose:** A tokenizer is established to convert words into numerical representations.
- **Parameters:** The maximum number of features is set to 15,000, embedding dimension to 512, and a maximum sequence length of 512 is defined.

Text Data Processing for Model Training and Evaluation:
In the preprocessing stage of our model training, the train data undergoes tokenization and padding operations. The purpose of this operation is to turn words into sequences of indices, and subsequent padding guarantees that all sequences keep a constant length of 512. A similar tokenization and padding approach is used to the evaluation dataset, where words are turned into sequences, and padding assures consistent sequence lengths. Extending this method to the test dataset, words in the test data are likewise turned into sequences, and padding is applied to preserve uniform sequence lengths. This consistent preprocessing strategy across training, evaluation, and test datasets provides compatibility, supporting successful model training and assessment.

Bidirectional LSTM Model Architecture Overview:

In this phase of our experimental design, we develop a neural network architecture using Keras, concentrating on a Bidirectional Long Short-Term Memory (LSTM) model. Here's a quick overview:

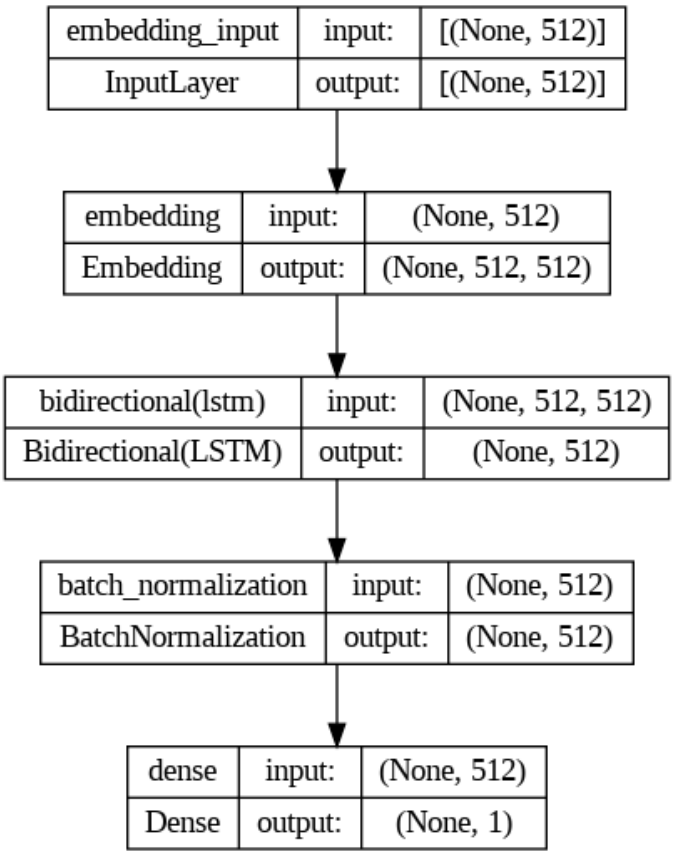


Figure 5.8: Model Plot

Embedding Layer:

1. Purpose: Generates dense word embeddings for input sequences.
2. Configuration: Vocabulary size set to 15,000 (`max_features`), embedding dimension set to 512 (`embedding_dim`), and input length set to 512 (`input_length`).

Bidirectional LSTM Layer:

1. Purpose: Captures bidirectional contextual information from input sequences.

2. Configuration: 64 LSTM units with a dropout rate of 0.3 for regularization.

Batch Normalization Layer:

1. Purpose: Normalizes the activations of the previous layer, aiding in the optimization process.

2. Configuration: Batch normalization layer introduced to enhance model stability.

Dense Output Layer:

1. Purpose:
   A fully connected layer with sigmoid activation for binary classification.

2. Configuration:
   Single output unit with sigmoid activation.

Model Compilation and Visualization:
In this stage of our experiment, the Bidirectional LSTM model is compiled, metrics are defined, and the model architecture is visualized.

Metrics and Other Parameters:

Metrics: BinaryAccuracy, F1, and Recall.
Loss Function: Binary cross-entropy loss.
Optimizer: Adam optimizer.
Epochs: 500
Patience: 3

ii. Experimental Setups for Rabindranath Era vs. Current Era

Author Labeling:
In order to boost the success of our categorization efforts, a modified categorical labeling technique has been designed, integrating the addition of an 'Author_label' column to the DataFrame. This labeling approach includes mapping individual authors to numerical values, hence improving the model's capacity to quickly spot patterns throughout the training phase. In this modified technique, writers such as Rabindranath Tagore, Bankim Chandra, Sharat Chandra Chattapaddhay, and Begum Rokeya have been awarded a label of 1, highlighting their similar features. Conversely, writers like Humayun Ahmed, Zafor Iqbal, and Ahmad Sofa have been awarded a label of 0, showing their different authorial features. This comprehensive method of author identification tries to identify and use inherent patterns within the material, adding to a more nuanced understanding throughout the model training phase.
Undersampling:

1. Train Data:

    - During the undersampling process, the dataset underwent balancing by establishing subsets for each author type. More exactly, 4589 examples were picked for category 1, 4589 for category 2, 4589 for category 3, 720 for category 4, 5080 for category 5, 4944 for category 6, and 5140 for category 7 using random sampling or replacement. The final train_data DataFrame amalgamates these subsets, guaranteeing appropriate representation for each author category. This comprehensive strategy boosts the model's generalization throughout both the training and evaluation stages.

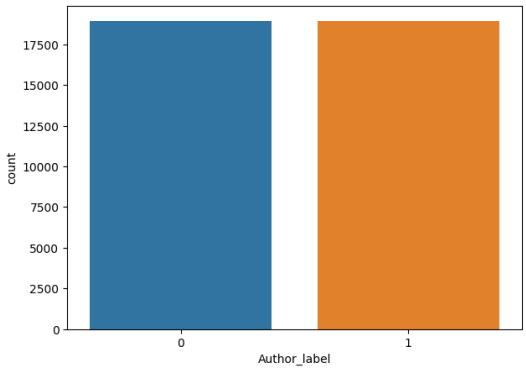


Figure 5.9: Train Data

2. Test Data:

    - During the undersampling phase of our experimental design, the major goal was to remedy the unequal distribution of authors within the dataset. This process entailed the construction of unique subgroups for each author type, changing the size of these subsets to fix the imbalance. In some categories, random selection with replacement was applied to

boost the number of minority authors. The merger of these subsets resulted in the construction of both the training and evaluation datasets, guaranteeing a more fair distribution of cases across different author categories. Subsequently, a similar strategy was utilized to build the test dataset. Specifically, 1130 instances from df31, 1131 instances from df32, 1131 instances from df33, 100 instances from df34, 1214 instances from df35, 1046 instances from df36, and 1232 instances from df37 were utilized to produce the test data subsets.. This thorough selection method guarantees a balanced representation for an impartial evaluation of the model's performance. The shuffling of examples across various datasets sought to generate unpredictability, boosting the model's power to generalize successfully among varied authors. This undersampling strategy creates the framework for a more strong and unbiased examination of our model.

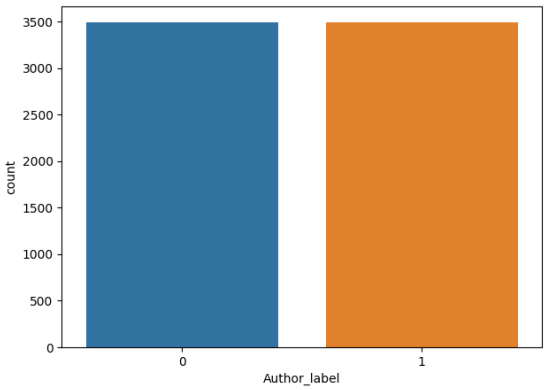


Figure 5.10: Test Data

Model Configuration (BERT + Bidirectional LSTM):

In this experimental work, we incorporate BanglaBERT, a transformer model especially constructed for tackling linguistic problems in the Bengali language. Associating with the Hugging Face Transformers library, we structure the BanglaBERT model and tokenizer to support contextual text embeddings. Besides, the inclusion of the 'normalizer' module is fine-tuned the text data to ensure its coherence. This strategy matches with recent developments in Natural Language Processing (NLP), certifying our model adeptly handles Bengali text. The inclusion of BanglaBERT fortifies our experimental architecture, offering a stable base for the successful deployment of NLP applications.

Data Preparation for BanglaBERT Integration: The distinct preparation of input data for both the training and testing stages is crucial. Engaging the BanglaBERT tokenizer, sentences experience encoding, leading to the generation of input IDs and attention masks to highlight key tokens. The dataset is partitioned into discrete training and testing subsets, giving arrays including input data (X_train_input and X_test_input), related labels (Y_train_label and Y_test_label), and attention masks (train_mask and test_mask). The subsequent application of a train-test split to these arrays is critical for the proper training and testing

of our model. These preliminary methods accelerate the smooth integration of BanglaBERT into our experimental pipeline, hence paving the path for competent execution in natural language processing jobs.

Application of Padding Sequences on Texts: In the context of Bidirectional Long Short-Term Memory (LSTM), the application of padding sequences entails the insertion of specific tokens or components to input sequences, assuring uniform length. This is especially crucial since LSTMs demand fixed-size inputs, but natural language processing (NLP) applications often involve text sequences of various lengths. The `'pad_sequences'` function, accessible in a selected library, effectively pads the sequences to ensure consistent length, a critical requirement for the training of neural networks. Following the construction of pad_sequences for our train, validation, and test texts, we proceed towards the building of our Bidirectional Model.

Model Architecture:

In this part, we design a Bidirectional LSTM model for the goal of author classification, including BanglaBERT embeddings. The model architecture is outlined as follows:

Input Layer: The model accepts input sequences with a maximum length of max_len tokens.

BanglaBERT Layer: Leveraging the BanglaBERT model, input sequences and attention masks are supplied to record contextual embeddings.

Bidirectional LSTM Layer: A Bidirectional Long Short-Term Memory (LSTM) layer is presented, containing 64 units, a dropout rate of 20%, and recurrent dropout of 20%. This layer helps the model to collect bidirectional contextual information from the BanglaBERT embeddings.

Output Layer: A dense layer with a softmax activation function generates probabilities for binary classification, expressing the probability of each author. The model seeks to categorize input sequences into one of the two author groups.

Model Setup:

In the model setup and compilation step, we have established particular parameters that guide the training process. Here's an explanation of the metrics and other parameters:

Loss Function: Sparse Categorical Crossentropy. The selected loss function is designed to do tasks including multi-class classification, where each case is assigned to a particular class. It computes the cross-entropy by measuring the gap between true labels and expected probabilities. It benefits effective training by penalizing departure from the actual class.

Metrics: Sparse Categorical Accuracy. TIn terms of metrics, it is enrolled to evaluate the model's prediction accuracy. It is suitable in multi-class classification scenarios as the metric provides valuable insights on the percentage of properly categorized occurrences.

Optimizer: Adam Optimizer. This optimization technique adjusts the learning rates of individual parameters independently which makes it a widely chosen method for training deep neural networks. Its efficiency in achieving convergence offers adaptability to various learning rates.

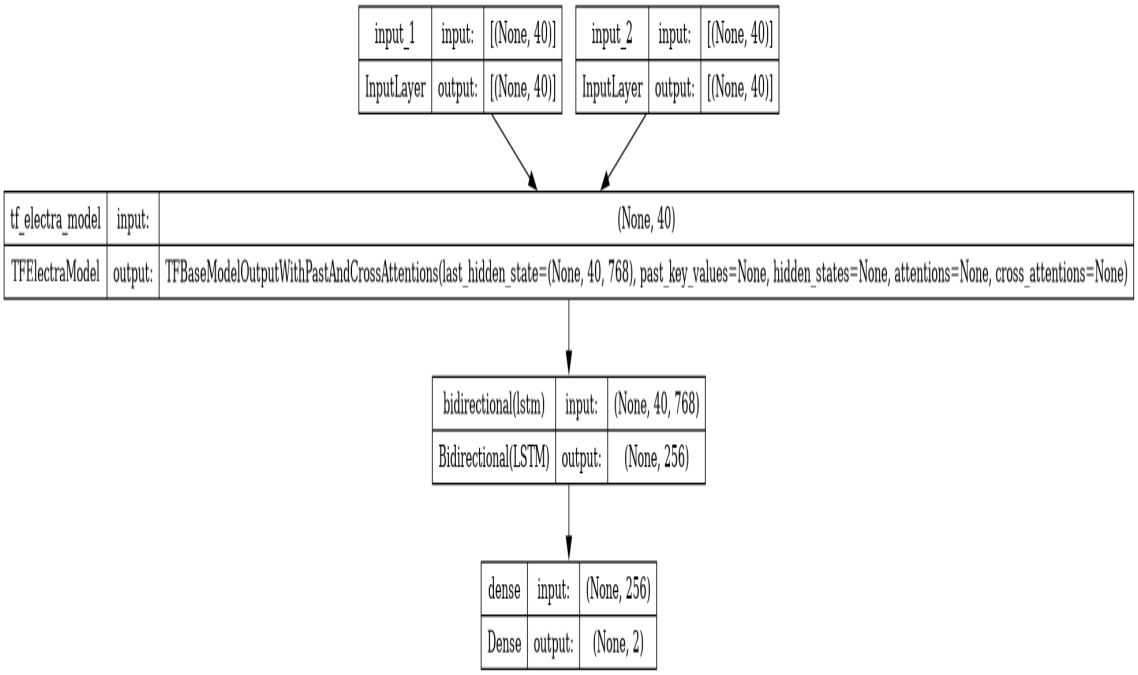


Figure 5.11: Model Plot

Model Fitting:

1. **Batch Size:** Each training batch is built of 64 samples (batch_size=64). Employing this batchwise processing increase is used to optimize memory utilization and accelerate the training process.

2. **Epochs:** The model undergoes several iterations over the entire training dataset, named epochs. In this case, the training process is executed for 500 epochs (epochs = 500), which allows the model to progressively enhance its performance.

3. **Early Stopping:** The EarlyStopping callback is introduced here to stop training if there is no significant improvement in the validation loss after one epoch (patience=3). This mechanism helps to avoid unnecessary training and refine the model's efficiency. The min_delta option determines the lowest change in the measure that is tracked to qualify as an improvement.

Metrics and Other Parameters:

Metrics: BinaryAccuracy, F1, and Recall.
Loss Function: Binary cross-entropy loss.
Activation Function: Sigmoid.
Optimizer: Adam optimizer.
Epochs: 500.

Model Configuration (Bidirectional LSTM)

Train + Evaluation Split:

- **Purpose:** Preparing the combined dataset for further splitting.
- **Operation:** Utilizing the `train_test_split` function to assign 80% of the data for training and 20% for assessment.

Textual Data Preparation for Model Input: In this part of our experiment, we apply tokenization and padding methods to prepare the textual data for effective model input.
Tokenizer Configuration:

- **Purpose:** A tokenizer is constructed to translate words into numerical representations.
- **Parameters:** The maximum number of features is set to 15,000, embedding dimension to 512, and a maximum sequence length of 512 is established.

Train Data Processing:

- **Purpose:** The training data undergoes tokenization and padding.
- **Operations:** Words are translated into sequences of indices, and padding guarantees that all sequences have a constant length of 512.

Evaluation Data Processing:

- **Purpose:** Similar tokenization and padding are done to the evaluation dataset.
- **Operations:** Words in the evaluation data are translated into sequences, and padding guarantees constant sequence lengths.

Test Data Processing:

- **Purpose:** Extending the tokenization and padding procedure to the test dataset.
- **Operations:** Words in the test data are turned into sequences, and padding guarantees uniform sequence lengths.

Output Dimensions:

- **Output:** The generated tensors for training, evaluation, and test data are presented, displaying the uniform dimensions attained using tokenization and padding.

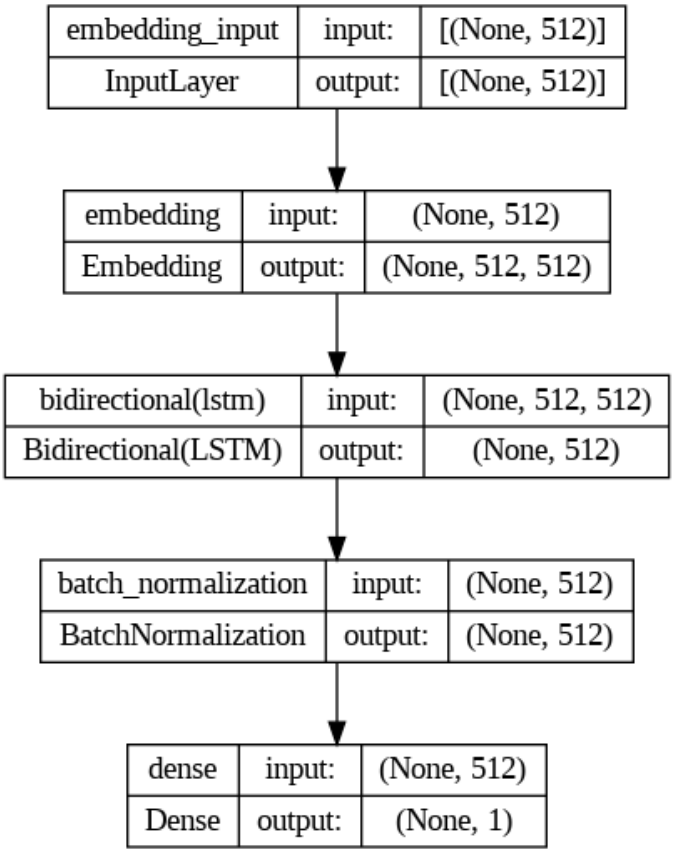


Figure 5.12: Model Plot

Bidirectional LSTM Model Architecture Overview: In this phase of our experimental design, we develop a neural network architecture using Keras, concentrating on a Bidirectional Long Short-Term Memory (LSTM) model. Here's a concise overview:

Embedding Layer:

- **Purpose:** Generates dense word embeddings for input sequences.
- **Configuration:** Vocabulary size set to 15,000 (`max_features`), embedding dimension set to 512 (`embedding_dim`), and input length set to 512 (`input_length`).

Bidirectional LSTM Layer:

- **Purpose:** Captures bidirectional contextual information from input sequences.
- **Configuration:** 64 LSTM units with a dropout rate of 0.3 for regularization.

Batch Normalization Layer:

- **Purpose:** Normalizes the activations of the previous layer, aiding in the optimization process.
- **Configuration:** Batch normalization layer introduced to enhance model stability.

Dense Output Layer:

- **Purpose:** A fully connected layer with sigmoid activation for binary classification.
- **Configuration:** Single output unit with sigmoid activation.

Model Compilation and Visualization: In this stage of our experiment, the Bidirectional LSTM model is compiled, metrics are defined, and the model architecture is visualized.

Metrics: Binary Accuracy, F1 Score, Recall
Loss Function: Binary Cross-Entropy Loss
Optimizer: Adam Optimizer
Activation Function: Leaky ReLU
Epochs: 500

iii. Experimental Setups for Adversarial Attack Classification Model

Tokenization method for Textual Data: In the earliest phases of our experimental setup, a rigorous tokenization method was performed to turn textual data into numerical indices, a key procedure in natural language processing. The configuration includes specifying crucial parameters:

1. **Maximum Number of Words (MAX_NB_WORDS):**
   Set at 50,000, selecting the most common terms to capture important language trends.

2. **Maximum Sequence Length (MAX_SEQUENCE_LENGTH):**
   Fixed at 512, maintaining consistency in the length of each text sequence for successful model processing.

3. **Embedding Dimension (EMBEDDING_DIM):**
   Established at 512, establishing the dimensionality of the created word embeddings.

Train + Evaluation Split:
In the essential phase of data preparation for model training and assessment, The train_test_split' function was used to accomplish train-test split. The 'X' and 'Y' variables, the features and labels respectively in train train and test sets ((X_train ¸ Y_train, Xtest , ytest)). It was set at a split ratio 85% training and 15 % testing, providing a broad range of data representation. 42 as the 'random_state' input assures repeatability of split. The Efficient data partitioning for future model training and assessment. **Textual Data Preparation for Model Input:**
In this phase of our experiment, we employ tokenization and padding techniques to prepare the textual data for effective model input. In this part of our experiment, we apply tokenization and padding methods to prepare the textual data for effective model input. **Tokenizer Configuration:**

1. **Purpose:**
   A tokenizer is developed to transform words into numerical representations.

2. **Parameters:**
   The maximum number of features is set to 15,000, the embedding dimension to 512, and a maximum sequence length of 512 is established.

Train Data Processing:

1. **Purpose:**
   The training data undergoes tokenization and padding.

2. **Operations:**
   Words are translated into sequences of indices, and padding guarantees that all sequences have a constant length of 512.

Evaluation Data Processing:

1. **Purpose:**
   Similar tokenization and padding are done to the evaluation dataset.

2. **Operations:**
   Words in the evaluation data are translated into sequences, and padding guarantees constant sequence lengths.

Test Data Processing:

1. **Purpose:**
   Extending the tokenization and padding procedure to the test dataset.

2. **Operations:**
   Words in the test data are turned into sequences, and padding guarantees uniform sequence lengths.

Output Dimensions:

1. **Output:**
   The generated tensors for training, evaluation, and test data are presented, displaying the uniform dimensions attained using tokenization and padding.

Bidirectional LSTM Model Architecture Overview: In this section of our experimental design, we construct a neural network architecture using Keras, focusing on a Bidirectional Long Short-Term Memory (LSTM) model. Here's a concise overview: **Embedding Layer:**

1. **Purpose:**
   Generates dense word embeddings for input sequences.

2. **Configuration:**
   Vocabulary size set to 10,000 (max_features), embedding dimension set to 512 (embedding_dim), and input length set to 512 (input_length).

Bidirectional LSTM Layer:

1. **Purpose:**
   Captures bidirectional contextual information from input sequences.

2. **Configuration:**
   64 LSTM units with a dropout rate of 0.3 for regularization.

Batch Normalization Layer:

1. **Purpose:**
   Normalizes the activations of the previous layer, aiding in the optimization process.

2. **Configuration:**
   Batch normalization layer introduced to enhance model stability.

Dense Output Layer:

1. **Purpose:**
   A fully connected layer with sigmoid activation for binary classification.

2. **Configuration:**
   Single output unit with Sigmoid activation.

Model Compilation and Visualization: In this stage of our experiment, the Bidirectional LSTM model is compiled, metrics are defined, and the model architecture is visualized.

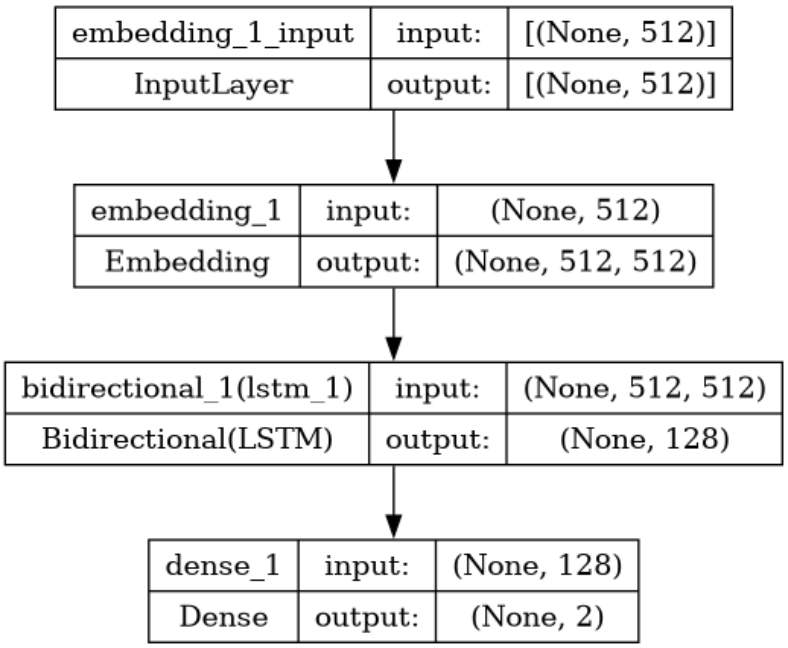


Figure 5.13: Model Plot

Model Training:
The model training process incorporates two essential callback functions, EarlyStopping and ModelCheckpoint, to enhance efficiency and ensure the best possible model is saved. Key components of the training process include:
Epochs and Batch Size:
The model is trained for 500 epochs with a batch size of 64, specifying the number

of times the entire training dataset is passed through the neural network.
Early Stopping Criteria:
EarlyStopping is implemented to monitor the validation loss. If the validation loss does not improve for three consecutive epochs ('patience=3'), training is halted early. This helps prevent overfitting and accelerates convergence.
Early Stopping Criteria:
EarlyStopping is implemented to monitor the validation loss. If the validation loss does not improve for three consecutive epochs ('patience=3'), training is halted early. This helps prevent overfitting and accelerates convergence.
Model Checkpointing:
ModelCheckpoint is utilized to save the weights of the best-performing model during training. The 'best_model.h5' file is updated only when a new minimum validation loss is achieved. This ensures the preservation of the most optimal model configuration.
Training Execution:
The 'fit' method is employed to train the model using the training data ('X_train' and 'Y_train'). The validation split is set to 10%, and both EarlyStopping and ModelCheckpoint callbacks are applied during training.
Metrics and other parameters:
Metrics: Binary Accuracy, F1 and Recall. Loss
function: Binary cross-entropy loss.
Optimizer: Adam optimizer.
Activation Function: Sigmoid
Epochs: 500

Model Configuration (BERT+Bidirectional LSTM)
Model Loading:
We load the total model into our setup from csebuetnlp. The model is compatible with TensorFlow, and it's using the TFAutoModel class. Additionally, the autotokenizer class is being used. The normalizer package was also loaded, giving capabilities for text normalization.
Data Preprocessing for Bangla BERT Model:
In the preprocessing stage of our research, we made many adjustments to the text data to enable the proper training and assessment of the Bengali BERT model. This included tokenizing the phrases from the training data using the Bengali BERT tokenizer, which included adding special tokens, restricting the sequence length to 40, and padding sequences as appropriate. As a consequence, we were able to build input IDs and attention masks for the training set. These were then turned into NumPy arrays, together with the required labels for training. The test words were subjected to comparable preprocessing methods, giving test input IDs and attention masks. Furthermore, the training set was partitioned into independent training and validation sets, namely 'X_train_input,' 'Y_train_label,' 'X_test_input,' and 'Y_test_label.' The appropriate attention masks were likewise distributed in the same way.
Bengali BERT-Based LSTM Model Architecture and Visualization:
In the present stage of our research, we are constructing a network structure using Keras, with an emphasis on a Bidirectional Long Short-Term Memory (LSTM) model. This network structure, built for author identification tasks, uses Bengali

BERT embeddings and merges the properties of Bidirectional LSTM layers. The model consists of two input layers, 'input_shape' and 'masks,' which serve as designated positions for the input sequences and accompanying attention masks. The Bengali BERT model, referred to as 'bmodel,' processes these inputs to produce embeddings, with the output retrieved from the first member of the resultant tuple. A Bidirectional LSTM layer follows, receiving bidirectional contextual information from the Bengali BERT embeddings. This layer has 100 LSTM units and features a 30% dropout rate to avoid overfitting. A densely linked layer with 2 units with 'leaky_relu' activation is utilized as the output layer for binary classification tasks. After specifying the architecture, the model is compiled to configure the loss function, optimizer, and evaluation metrics, ensuring the network is ready for optimum training. A representation of the model architecture is also given. The Sparse Categorical Crossentropy loss function is employed, appropriate for multiclass classification jobs where the labels are integers. The model's performance is assessed using the Sparse Categorical Accuracy measure, offering insight into the accuracy of predictions. The Adam optimizer with a learning rate of 2e-5 is used to optimize the model's weights during training. The 'compile' technique combines the supplied architecture, loss function, optimizer, and metrics, preparing the model for the training phase.

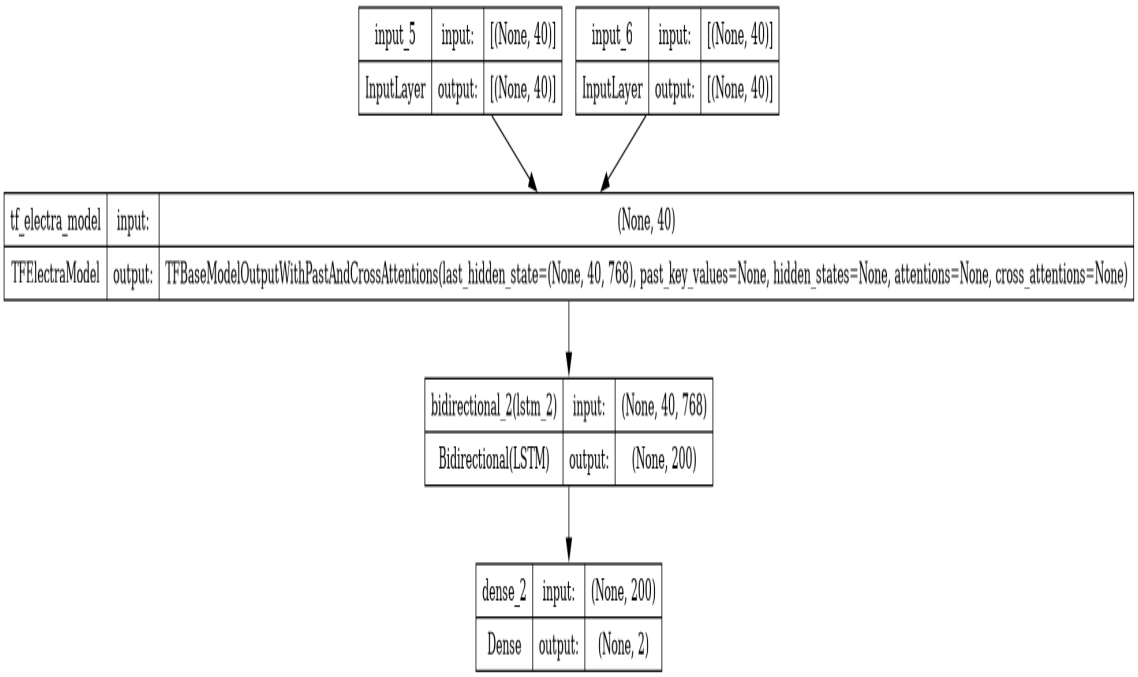


Figure 5.14: Model Plot

Metrics and other parameters: Metrics: Binary Accuracy, F1 and Recall.
Loss function: Binary cross-entropy loss.
Optimizer: Adam optimizer.
Activation Function: Softmax
Epochs: 500

iv. Experimental Setups for Rabindranath vs. Multiple Authors:

Model Configuration (Bidirectional LSTM)
Author-Balanced Data Selection:
In order to counterbalance the imbalance in the dataset to assure balance , a systematic undersampling strategy is implemented ,depiction of writers. Key steps in this process include:

Author-particular Data Extraction:
Three particular datas are taken out. Poets like Rabindranath Tagore, and Sharat Chandra. Separate data frames (df1, df2 and so on.) are established for each author.

Undersampling:
For each author, a random subset is picked in order to establish a balanced distribution. Each author's sample size is set at 40,070. replacement is permitted in order to preserve consistency. The produced data frames Balanced subsets for each author are (df5, df6, df7).

Tokenization and Padding:
The preprocessing stage consists of a tokenizing and padding network.

Tokenizer Configuration:
'Tokenizer' class is specialized using parameters such as numWords ((maximum number of words to utilize), "filters" (characters to filter) 'lower'. This configuration is tailored to the peculiarities of Bengali script.
Tokenization:
The tokenized texts are supplied into the tokenizer 'bangla data' dataset. It produces a vocabulary index ("word index") according to 50, the most prevalent terms; assigns unique indices to each of them.

Padding Sequences:
The tokenized sequences are then padded to a specified. 250 tokens long sequences using 'pad sequences' method.

Data Tensor form:
The resultant data tensor ('X') has a form that represents the padded sequences, making it acceptable for input into the neural network. The print statement validates the form of the data tensor.

Label Tensor Formation:
The labels ('Author') are one-hot encoded using 'pd.get_dummies,' forming a label tensor ('Y') with a shape that corresponds to the number of distinct authors.

Train-Test Data Split for Model Evaluation:
An essential step prior to model training and assessment entails splitting the preprocessed dataset into a set for training and another one for testing. The 'train_test_split' tool aids this division by putting aside 15% of the data particularly for testing purposes. 42 is chosen as the random seed to guarantee that it may be reused for various runs. These resultant sets identified as 'X_train' , 'X_test', 'Y – train, '

and are suggestive of the training and testing data together with label relations. The print statements validate these sets' size, therefore offering a glimpse into the amount and type of data that will be utilized to train This exact split offers an impartial evaluation of the model's capacity to generalize on unknown input during future testing.

Bidirectional LSTM Model Architecture Overview:

The model of the author classification is constructed using a Sequential architecture in Keras , using core layers to parse text fast. The model is launched with an Embedding layer, converting discrete words into continuous vectors that capture semantic relationships. 64 LSTM units with a 30% dropout rate for regularization are employed on Bidirectional Long Short-Term Memory (LSTM) layer to boost the model's capacity to learn context and dependencies in both directions. So, the output layer contains a Dense Layer with sigmoid activation function for multi-class classification and 'adam' optimizer as well as categorical_crossentropy is utilized to design this model. The summary printout offers a detailed perspective of the architecture, illustrating how effectively it could evaluate sequential data and detect tiny patterns accurately. This matches to what author categorization aspires for.

Model Compilation and Visualization:

In this stage of our experiment, the Bidirectional LSTM model is compiled, metrics are defined, and the model architecture is visualized.

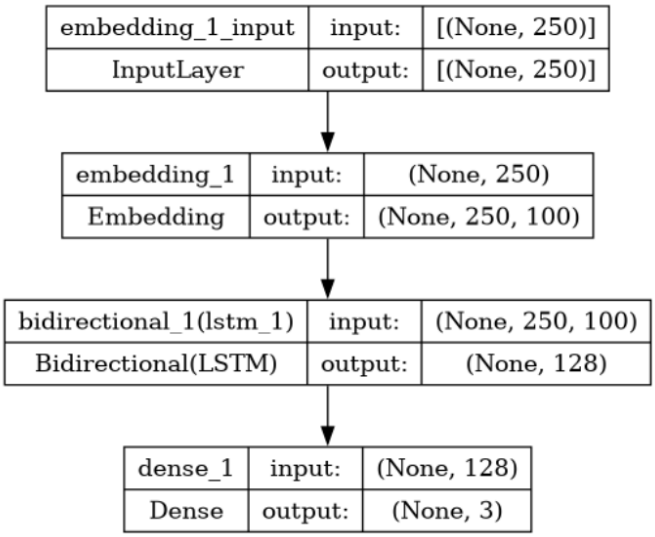


Figure 5.15: Model Plot

Training Strategy with Early Stopping:
The training technique for the Bidirectional LSTM author classification model comprises two critical callbacks, namely EarlyStopping and ModelCheckpoint. EarlyStopping is designed to monitor the validation loss ('val_loss') and intervenes if there is no improvement for three consecutive epochs, with a minimum loss change below a given threshold. This proactive strategy minimizes overfitting and increases training efficiency. Simultaneously, the ModelCheckpoint callback guarantees that the weights of the best-performing model are preserved throughout training. The stored model, entitled 'best_model.h5,' is selected by minimizing the validation loss, assuring that the model with optimum generalization on the validation set is kept. This deliberate mix of callbacks refines the training process, boosting model efficiency and resilience. The training script runs throughout 500 epochs with a batch size of 64, providing a complete approach to model training and preservation.

Metrics and other parameters:
Metrics: BinaryAccuracy, F1 and Recall.
Loss function: Binary cross-entropy loss.
Activation Function: Sigmoid
Optimizer: Adam optimizer.
Epochs: 500

Model Configuration (BERT + Bidirectional LSTM)

BanglaBERT Integration:
The state of the BanglaBERT model was loaded into our test setup using the TFAutoModel method. The AutoTokenized is also used along with the model. This is a pretrained language model which is trained with complexity of Bangla.

BanglaBERT Tokenization:
The initial stage in authorship attribution preparation is tokenizing the Bengali text data by using BanglaBERTtokenizer. Setting max_len to 40, the training data phrases are processed in order to create IDs for inputs and masks connected with attention. Using the encode_plus function from BanglaBERT tokenizer phrases are encoded, special tokens added, making sure that it does not exceed 40 and adding attention masks. Generate input IDs, attention masks, and matching author labels are grouped in training. This is similarly done to the test data which creates input IDs, attention masks and also test. The training and test datasets are then separated into input data, labels and attention mask so as to allow continued training of the authorship attribution model. This tokenization stage guarantees that the Bengali text is appropriately prepared up for integration into BanglaBERT in the machine learning pipeline.

BanglaBERT Model Architecture:
In this approach, LSTM neural network is combined with BanglaBERT embeddings to produce a good model for authorship attribution. create_model function describes the structure of a neural network. 64 units across Bidirectional LSTM layer and dropout rates of 0.2 are used to capture bidirectional relations among input sequences. The final completely dense layer, with softmax activation function

creates probability scores of each class involved for multi-class authorship attribution. The final completely dense layer, with softmax activation function creates probability scores of each class involved for multi-class authorship attribution. The final dense layer with a softmax Activation function is responsible for the output of probability probabilities per class so that multi-label authorship attribution may be carried out. Multi-class Mortality Attribution The final Link concludes with the dense layer which is filled and employs softmax activation for scoring probabilities to each class. This architecture utilizes the contextual representations supplied by BanglaBERT and sequential learning of Bidirectional LSTM to boost the model's capabilities in recognizing authorial styles. In general, the core of this model's summary gives insight about the configurations of layers and overall structure.

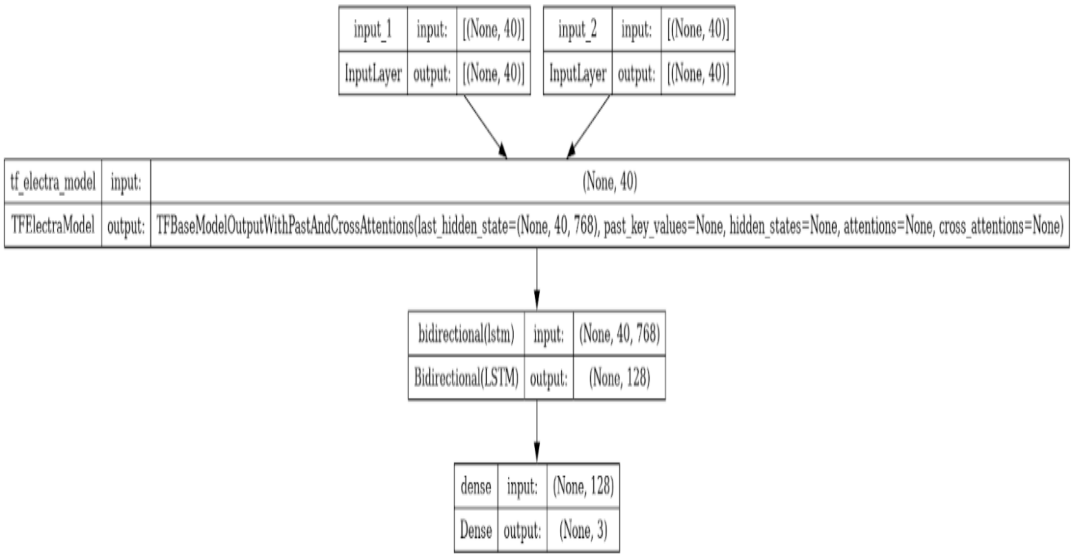


Figure 5.16: Model Plot

Metrics and other parameters:
Metrics: BinaryAccuracy, F1 and Recall.
Loss function: Binary cross-entropy loss.
Activation Function: Softmax
Optimizer: Adam optimizer.
Epochs: 500

5.2 Text generation in Rabindranath's Style

Methodologies that create text in the style of Rabindranath Tagore is a challenging task as it tries to write in a specific manner that captures the essence and theme of the Bengali-renowned author's writing. We're using a T5 transformer model to help us do this. Rabindranath Tagore was a well-known writer and Nobel Prize winner known for his beautiful and profound writing. He had a special way of using words that were poetic and deeply connected to the culture and spirituality of Bengal. In this study, we're going to see if we can make a machine-learning model write like he did. Our goal is to create text that sounds like it could have come from Tagore himself, and in doing so, we hope to better understand his literary style and contribute to the field of text generation.

5.2.1 Models

T5 (Text-to-Text Transfer Transformer) stands out for its ability to perform a wide range of NLP tasks through a unified framework. As we transition into the next research phase, focusing on replicating an author's writing style, our thesis incorporates headfirst the Transformer-based T5 model for text generation to capitalize on its capabilities in advanced NLP. We utilize T5's pre-trained weights, fine-tuned to capture the unique intricacies of the author's typing style, in our implementation. Its unique "text-to-text" approach allows it to handle diverse tasks by converting them into a unified text generation problem. This methodology aligns with our objective of generating text that mirrors the author's unique writing patterns. During the fine-tuning process, the T5 model learns to understand the context to recognize and mimic specific patterns such as sentence structures, word choices, and even idiosyncrasies in the punctuation usage of the author's writing. In the fine-tuning phase, we let the model learn from a selective dataset, allowing it to refine its grasp of unique signature patterns (characteristics) in the author's written work. As a result, the T5 model stands out in generating text that effectively imitates the author's style as well as exhibits a noteworthy degree of context awareness.

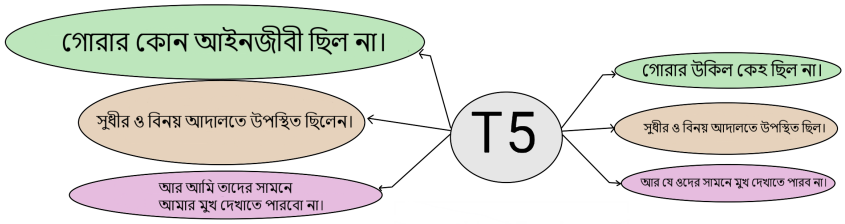


Figure 5.17: T5 Model

BanglaT5

BanglaT5 is a model that aligns with our project objective to classify by generating a specific author's unmatched writing style. Just like other T5 models, this is also another version of the base T5 model. This is a powerful T5 tool specially

designed for the Bengali language. We utilize the maximum potential of BanglaT5 to generate a piece of writing that is able to exactly imitate the style of an author by preserving the distinctive pattern of their writing, that too with high accuracy. As mentioned before, banglaT5 promises to increase the precision of our research via understanding and implementation of Bengali language to new heights, similar to the way T5 efficiently carries out NLP tasks such as authorship attribution. The particular approach of BanglaT5 matches effortlessly with our aims, helping us dive deeper into the core of how Bengali writers present themselves. This transition is not just about technology, it reflects our commitment to precision and cultural sensitivity that resonates beyond the confines of traditional English-centric methods. Through this advanced fusion of technology and language, we anticipate a more refined and contextually nuanced emulation of the author's writing style, showcasing how transformer models can keep getting better at figuring out who wrote what in the world of authorship attribution.

mT5

We have used the mT5 Model for Rabindranath-style text generation. The mT5 language model, otherwise known as the multilingual Translation Transformer model was developed by Google and mentioned in the "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer" paper in 2019, for multilingual translation tasks. It is also a variant of the T5 model or Text-to-Text Transfer Transformer Model. Since it is able to deal with various languages, thence the utility of mT5 spans over a large domain of linguistic contexts in NLP. mT5 is an enhanced version of the T5 model, hence, it is able to work with an increased number of parameters which tends to promote retrieval effectiveness. mT5 is a renowned adaptable tool which can efficiently carry out translation, language perception, and summarization.

Background of mT5: As we already know, T5 is a pre-trained language model characterized by its application of a single "text-to-text" format for all text-based NLP challenges. The most widespread usage of the T5 model is to perform generative tasks where the model generates text conditioned on some specific input [15]. T5 uses a basic encoder-decoder Transformer architecture. T5 is pre-trained on a "span-corruption" goal in masked language modeling, in which successive spans of input tokens are substituted with a mask token or is corrupted and the model is trained to reconstruct the masked-out words. This way, it learns to understand the contextual relationships and dependencies within the text. Now, we shall discuss a little about mT5 here. Unlike that of the T5 model, mT5 requires adequate fine-tuning before it can be used for any task. Furthermore, Google's mT5 was simply pre-trained on mC4 - a new Common Crawl-based dataset [15], with no supervised training covering 101 languages.

mT5-small

Multilingual T5 Small (mT5 Small) is a compressed variant of the T5 language model, and similar to mT5 this model is also used for multilingual natural language processing tasks. mT5 small is also able to work with and generate multiple languages as a result of being trained on a wide range of language corpuses. The

objective of building a small model such as this one is to implement it on applications with resource constraints without hampering its quality of performance and delivering valid output across multilingual contexts. However, despite its smaller size when compared to bigger alternatives, the mT5-Small has the ability of performing a variety of language-related tasks such as translation, summarization, and language interpretation.

Background of mT5-Small: T5 is a transformer-based paradigm that defines all NLP tasks as text-to-text problems, bringing together activities such as translation, summarization, and question-answering into a unified framework. The ideology of this compact variant of T5, named "mT5-Small", is quite well-known amongst transformer-based models. These more compact models balance computing demands and performance by being trained on a smaller amount of data or with fewer parameters. And the objective of researchers behind building such compact toned-down variants of the original model is to facilitate accessibility for such tasks that have resource constraints or to improve efficiency in specific applications.

Architecture Overview of mT5 Model:

Just like the original T5 Language model, the mT5 model and mT5-small model have similarities in their architecture. The main features of the architecture are described below:

1. **Encoder-Decoder Architecture:** Just like T5, mT5 has an encoder-decoder design. The encoder generates the output text from the input text, while the decoder reverses the process. Transformer designs are used in both the encoder and the decoder.

2. **Positional Embeddings:** To deal with the text's sequential structure, mT5 uses positional embeddings, which offer information about the position of each token in the sequence.

3. **Multi-Head Self-Attention:** Both the encoder and decoder transformer blocks are made up of multi-head self-attention mechanisms. This allows the model to focus on multiple sections of the input sequence at the same time, capturing dependencies at different points.

4. **Feedforward Neural Networks:** Following the self-attention layers, feedforward neural networks are used to process input independently at each point.

5. **Layer Normalization and Residual Connections:** Every sub-layer (such as attention and feedforward) is preceded by layer normalization and is linked by a residual connection, which aids in training stability and convergence.

6. **Vocabulary and Tokenization:** To turn input text into tokens, mT5, like similar transformer-based models, adopts a tokenization approach. Usually, the model is trained upon a fixed-size vocabulary that encompasses a broad spectrum of languages.

5.2.2 Experimental Setup of the used models

Setup for BanglaT5 and mT5-small

As mentioned earlier, text generation that imitates the style of Rabindranath Tagore is one of the focal tasks that completes our project. Below are the steps that were needed to successfully carry out the text generation part.

1. **Dataset Processing:**

   Our dataset contains two columns: 'Input_text' and 'Rabindranath_text'. Our main objective here is to provide a simple input text which shall be read and then converted into a writing that properly copies the writing personality/style of Rabindranath Tagore. Then, by using 'df.shape' we have found the exact number of data, which was 22899 texts. This data was then split into 3 parts: train_data, test_data, and eval_data. After symmetrical division, we created a csv file for each of the datasets for various purposes. The number of data are mentioned here: training data: 17000 sentences, test data: 3000 sentences and eval data: 2899 sentences.

2. **Loading Tokenizer Model:**

   The model that we are about to fine-tune, at first we will load its subsequent tokenizer from Hugging Face transformers library.

3. **Tokenization for Text Generation:**

   Next, specific columns of the DataFrame are extracted for the goal of tokenizing input and target texts using the Tokenizer. As a consequence, input and target tensors are made to represent the tokenized sequences of the input and target.

   This tokenization involves a specific process -

   ```
   inputs = tokenizer(input_texts, return_tensors='pt',
   padding=True,
   truncation=True,
   max_length=128)
   ```

   This line of code tokenizes the input_texts using the Tokenizer of the model being fine-tuned. The resulting tokens are converted to PyTorch tensors. It also includes padding, truncation, and limits the maximum sequence length to 128. Then we stored the tokenized input_texts in the variable inputs. Then,

   ```
   targets = tokenizer(target_texts, return_tensors='pt',
   padding=True,
   truncation=True,
   max_length=128)
   ```

   Similar to the previous line, it tokenizes the target_texts using the same tokenizer and parameters. Then we stored the tokenized target_texts in the variable targets.

4. **Declaration of Custom Dataset Loading Class:**

   The purpose of this specific class is to promote sequence-to-sequence training of neural networks. The intended application of CustomDataset is in relation to transformer-based models. The dataset is initialized using tokenized sequences as inputs and targets. It provides methods for determining the length of the dataset (___len___) and extracting individual items (___getitem___). The dataset returns a dictionary that includes $'input\_ids'$, $'attention\_mask'$, $'decoder\_input\_ids'$, $'decoder\_attention\_mask'$, and $'labels'$ for each item. Then, a custom dataset called Custom_Dataset is created using the earlier tokenized inputs and targets. Finally, a PyTorch DataLoader is created, which batches and shifts the data with a batch size of 8, making it great for training transformer models with mini-batch optimization. The DataLoader may be utilized in training loops for iterating over batches of data while training models.

   The ___getitem___ method returns a dictionary containing the necessary data for a single training sample. This includes:
   i. 'input_ids': Input sequence token IDs.
   ii. 'attention_mask': Attention mask for the input sequence.
   iii. 'decoder_input_ids': Target (decoder) sequence token IDs.
   iv. 'decoder_attention_mask': Attention mask for the target (decoder) sequence.
   v. 'labels': Target labels, which are the same as the decoder input IDs.

5. **Tokenization for Evaluation:**

   By using all the similar steps mentioned above we repeat this whole procedure for evaluation data. Initially, we read the dataset and separate the input text from the Rabindranath-style target text and tokenized the dataset. Then we create a class for the Evaluation Dataset where we sort all the data and finally store it in the Evaluation_Dataset.

6. **Model Loading and Training:**

   The model, loaded using the "$csebuetnlp/banglat5$" for banglaT5, and for mT5 small, $google/mt5-small$ pre-trained weights are used for sequence-to-sequence tasks. The model is set to training mode using model.train(). This step is important if we want to fine-tune the model on a specific job using our own dataset. By running train(), we enable the model to adjust its parameters during the training phase.

7. **Training of Model: (Metrics & Parameters)**

   (a) We have passed the tokenizer and previous model in the data_collater. The data collator is used to pre-process batches of data before they are fed into the model during training. It takes a batch of samples and tokenizes and adds paddings to them to make sure that all sequences in the batch are the same length, which is required for effective training with mini-batches. It also handles different arguments according to the Model.

   Some code parameters include:

```
per_device_train_batch_size=8:
per_device_eval_batch_size=8:
num_train_epochs=200:
evaluation_strategy='steps':
eval_steps=500:
save_total_limit=2:
load_best_model_at_end=True:
```

(b) Create a Trainer Instance:

The Trainer is configured with the pre-trained model (model), training arguments (training_args), a data collator for sequence-to-sequence tasks (data_collator), and training and evaluation datasets (custom_dataset and evaluation_dataset, respectively) to configure various aspects of the training process, such as batch size, epochs, output directory, etc. Additionally, it contains callbacks=[EarlyStoppingCallback(early_stopping_patience=3)], an EarlyStoppingCallback as a callback with patience of 3, allowing for early stopping during training if the evaluation measure does not improve for subsequent epochs. The trainer. train() function is then used to commence the training process, fine-tuning the model on the given datasets according to the defined training parameters and stopping conditions.

We required only 12 epochs to train the BanglaT5 model even though we had declared 200 epochs earlier due to early stopping. Similarly, for mT5-small, the number of epochs required was 32 only.

8. **Saving the Model:**

The `model.save_pretrained('fine_tuned_model')` stores the fine-tuned model parameters, settings, and any other files connected with the model in a directory named `'fine_tuned_model'`.

Similarly, the `tokenizer.save_pretrained('fine_tuned_model')` stores the tokenizer's vocabulary and configuration in the same directory. This enables for easy recovery and reuse of the fine-tuned model and tokenizer for future tasks or deployment without having to retrain or tokenize from start.

Chapter 6

RESULT AND ANALYSIS

6.1 Classification

6.1.1 Experimental findings from Rabindranath vs. Non Rabindranath Text Classification

Bidirectional LSTM: From our Bidirectional LSTM model for the classification of Rabindranath and non- Rabindranath texts, we obtained a sufficiently good accuracy of 91

Classification Report				
	Precision	Recall	F1-Score	Support
0	0.91	0.92	0.92	15204
1	0.92	0.91	0.92	15204
Accuracy	–	–	0.92	30408
Macro Avg	0.92	0.92	0.92	30408
Weighted Avg	0.92	0.92	0.92	30408

Table 6.1: Accuracy Report of Bidirectional LSTM for Rab vs. Non Rab data

The image shows a table of precision, recall, F1 score, support, accuracy, macro average, and weighted average of the Bidirectional LSTM Model used to evaluate the Rabindranath vs. non Rabindranath data. Here,Rabindranath Tagore means 1 and rest of the authors means 0.
According to the table, we can see that the model performs quite well overall with an accuracy of 0.92. Moreover, we also see that the macro average f1 score is also 0.92, which is significantly good. One advantage of this model is that there is no class imbalance present, as both the classes '0' and '1' have the same support value, which is 15204. Their f1-scores and recall values are also the exact same which proves the 'no class imbalance' point. Anf finally, the weighted average F1 score is 0.92, which is found by considering both classes together, indicating that the model is performing well on both classes. Overall, the image shows that the model has good performance on the classification task. Thus, it will be easier to interpret the results accordingly.

Evaluation of Bidirectional LSTM Model:

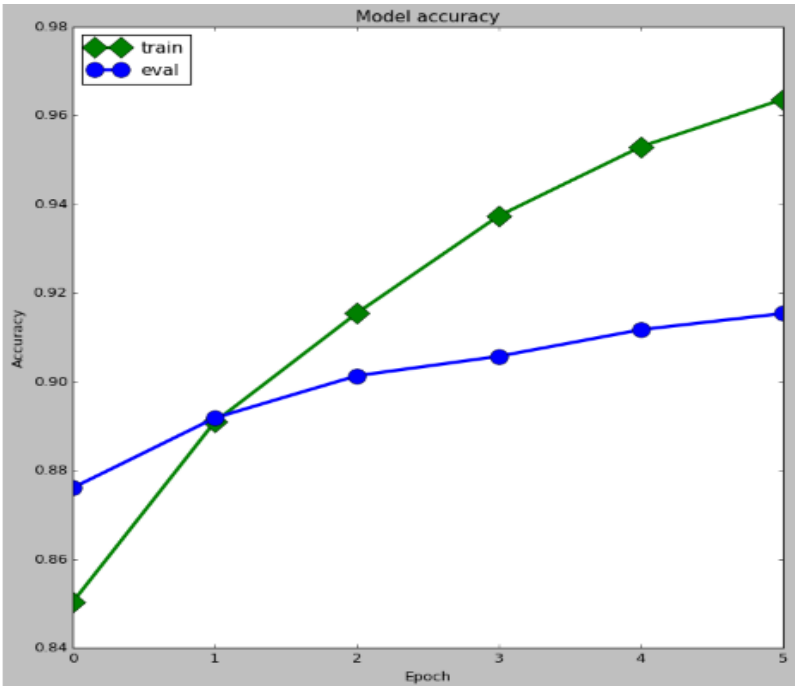


Figure 6.1: Epoch vs Accuracy

Here we can see that the accuracy of the model increases over time, reaching a peak of nearly 92% at epoch 5.0. This suggests that the performance of the model during evaluation was quite good. This graph also displays that the model is learning and improving over time. The initial increase in accuracy is likely due to the model finding patterns in the data. After epoch 1.0, the increase in accuracy decreases as we can see a less slanted gradient. This may be due to the model overfitting to the training data. Overall, the graph shows that the model is performing well and is able to achieve high accuracy on the test data.
It is important to note that the model accuracy on the held-out test set is a more accurate measure of the model's generalization performance than the model accuracy on the training set. This is because the test set contains data that the model has never seen before, so it is not able to overfit to the training data.

The graph shows that the loss of the model decreases over time. This graph shows that the model is learning and improving over time. The initial decrease in loss is

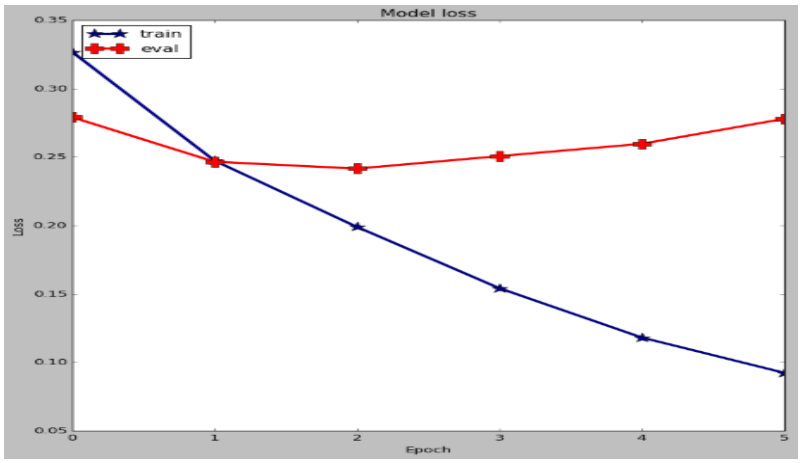


Figure 6.2: Epoch vs Loss

likely due to the model finding patterns in the data and the subsequent increase in loss may be due to the model overfitting to the training dataset.
Overall, the graph shows that the model is performing well and is able to achieve low loss on the test data.

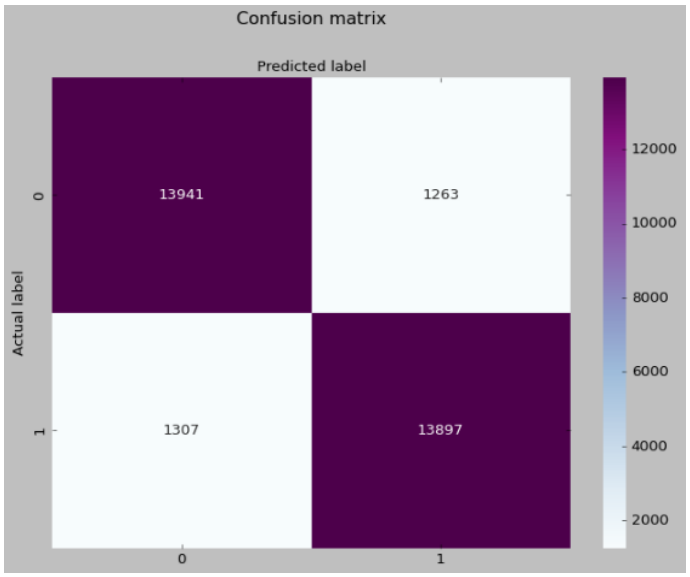


Figure 6.3: Confusion Matrix

Here, the confusion matrix helps us understand how well the model is actually working. As we gave 2 classes in our classification, we have a 2x2 matrix with the support values given in the matrix. We can calculate the False Positive (FP), False Negative (FN), True Positive (TP), True Negative (TN) values from this graph. After analyzing the graph, It has the following values - True Positives (TP): 13941, False Positives (FP): 1263, False Negatives (FN): 1307 and True Negatives (TN): 13897.

From the above graph we can get a good idea of the overall performance of our model. Here, the y-axis represents the True Positive value rate of our model and the x-axis shows the False Positive rate.

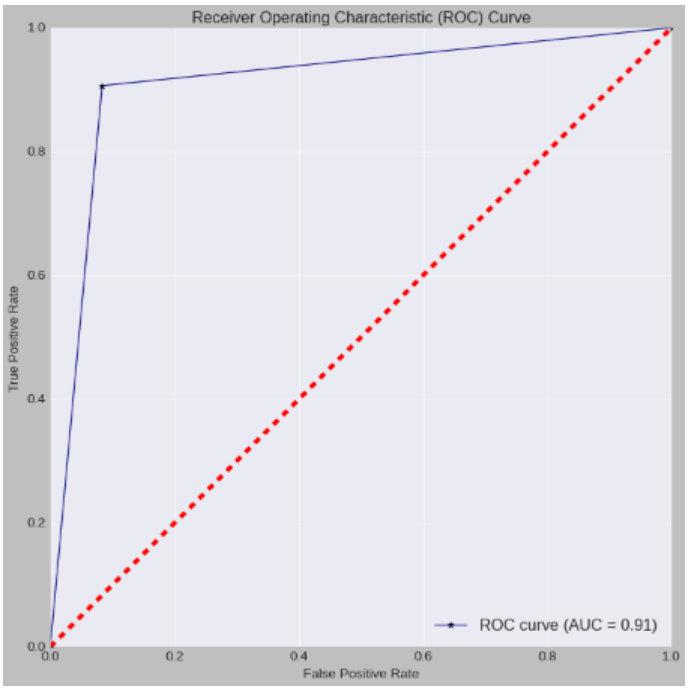


Figure 6.4: ROC Graph

The ROC curve is close to the top-left corner of the graph, which is an indication of good performance. The curve starts off rising steeply, which means that the model is very good at correctly identifying true positives when the threshold is low. As the threshold increases, the curve rises less steeply, but it remains well above the diagonal line. This means that the model is still able to perform well even when it is more cautious about classifying cases as positive. Then, the AUC (Area Under the Curve) value for this ROC curve is 0.91, which is a very good score. This indicates that the model is very good at distinguishing between positive and negative cases.

BanglaBERT + Bidirectional LSTM:

Classification Report				
	Precision	Recall	F1-Score	Support
0	0.95	0.96	0.95	15204
1	0.96	0.95	0.95	15204
Accuracy	–	–	0.95	30408
Macro Avg	0.95	0.95	0.95	30408
Weighted Avg	0.95	0.95	0.95	30408

Table 6.2: Accuracy Report of Bidirectional LSTM for Rab vs. Non Rab data

The report summarizes the model's performance on a test dataset, breaking down its predictions for each class into several metrics such as - precision, recall, F1 score, support, accuracy, macro average, and weighted average for the BanglaBERT + Bidirectional LSTM Model used to evaluate the Rabindranath vs. non Rabindranath

data. Here,Rabindranath Tagore means 1 and rest of the authors means 0.
The bottom part of the table shows that the model has an overall good performance. The table's bottom displays an accuracy score of 0.95, reflecting the model's ability to correctly predict outcomes across both classes. Additionally, it stands on macro average F1 score of 0.95. Notably, there is no class imbalance as both '0' and '1' classes have an identical support value of 15204, signifying an even distribution of samples. Class 1 exhibits impressive precision, recall, and F1-score, all hovering around 0.96, indicating the model's strong capability in correctly identifying instances belonging to this class. Meanwhile, class 0 also demonstrates commendable performance, with precision, recall, and F1-score approximately at 0.94.
This classification report suggests that the model is performing well overall on this dataset. It is able to accurately identify both positive and negative cases with high precision and recall.

Evaluation of BanglaBERT+Bidirectional LSTM Model:

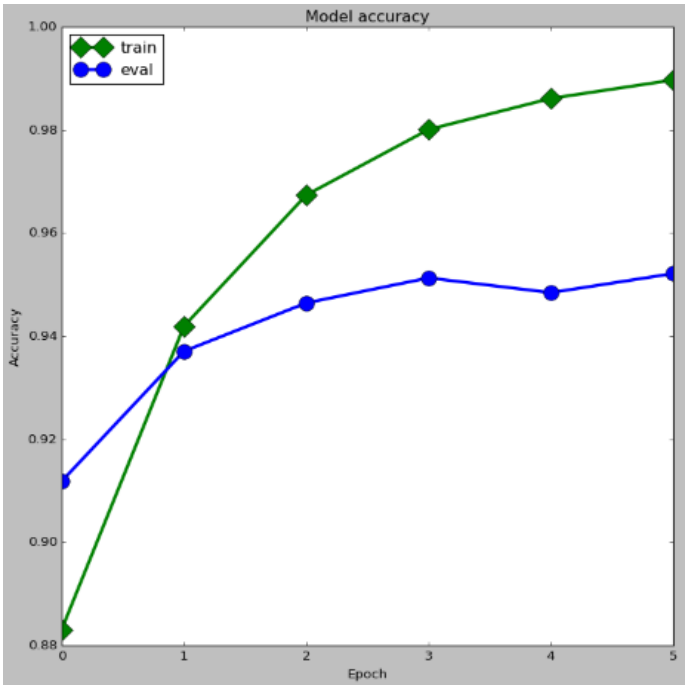


Figure 6.5: Epoch vs Accuracy

Here we can see that the accuracy of the model increases over time, reaching a peak of about 0.94 at epoch 1.0 and then plateaus. Still the value is near 0.95. This suggests that the performance of the model during evaluation was quite good.
This graph also displays that the model is learning and improving over time. Initially, the increase in accuracy can be explained to the model identifying patterns in the data. A flatter or steep gradient indicates that the pace of accuracy, which was increasing, slows down after the first epoch. This may be a sign of overfitting, which occurs when a model adapts to the training set too much. To sum up, the visual illustration indicates that the model operates well and achieves notable accuracy

levels on the test dataset. However, the plateau we see in the graph suggests that the model is unable to significantly improve performance after a certain point.

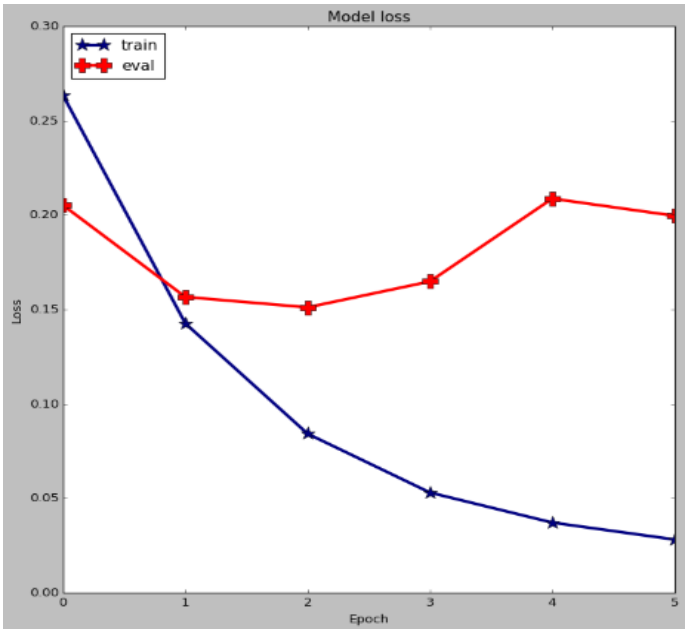


Figure 6.6: Epoch vs Loss

The graph shows that the loss of the model decreases over time, reaching a minimum of about 0.15 at epoch 1.0. After that point, the loss starts to increase up to 0.20. This graph indicates that the model is learning and improving as time progresses. The initial drop in loss can probably be attributed to the model recognizing patterns in the data, and the subsequent rise in loss might occur because the model is becoming too specialized in fitting the training dataset.
Overall, the graph shows that the model is performing well and is able to achieve low loss on the test data.

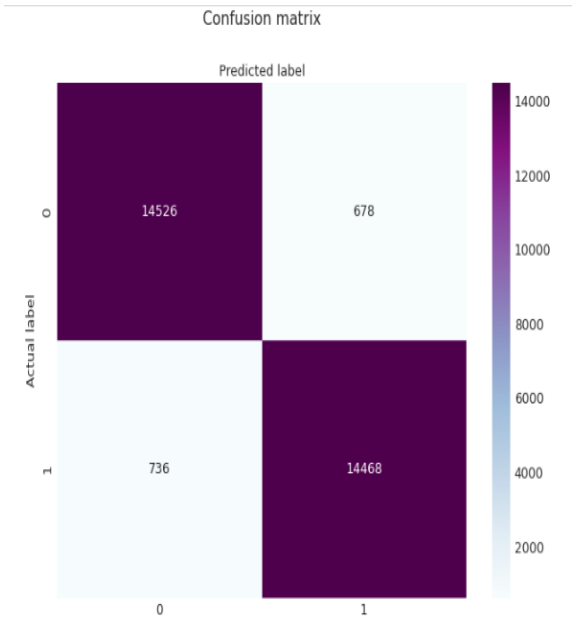


Figure 6.7: Confusion Matrix

Here, the confusion matrix helps us understand how well the model is actually working overall. Here, we can calculate the False Positive (FP), False Negative(FN) , True Positive (TP) and True Negative (TN) values from this graph. After analyzing the graph, It has the following values - True Positives (TP): 14526, False Positives (FP): 678, False Negatives (FN): 736 and True Negatives (TN): 14468.

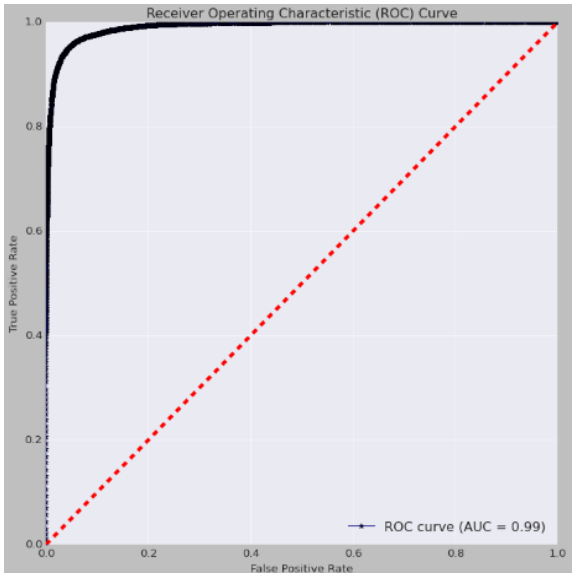


Figure 6.8: ROC Curve

From the above graph we can get a good idea of the overall performance of our model. Here, the y-axis represents the True Positive value rate of our model and the x-axis shows the False Positive rate.
The ROC curve is close to the top-left corner of the graph, which is an indication of good performance. The curve starts off rising steeply, which means that the model is very good at correctly identifying true positives when the threshold is low. As

the threshold increases, the curve rises less steeply, but it remains well above the diagonal line. This means that the model is still able to perform well even when it is more cautious about classifying cases as positive. Then, the AUC (Area Under the Curve) value for this ROC curve is 0.99, which is a very good score. This indicates that the model is very good at distinguishing between positive and negative cases. Furthermore, at the point on the curve where the sensitivity is 0.8, the specificity is about 0.95. This means that the model is able to correctly identify 80% of the true positive cases and 95% of the true negative cases. This is a good balance between the two metrics.

6.1.2 Experimental findings from Rabindranath Era vs. Current Era text classification:

Bidirectional LSTM:

Classification Report				
	Precision	Recall	F1-Score	Support
0	0.91	0.95	0.93	3492
1	0.95	0.91	0.93	3492
Accuracy	–	–	0.93	6984
Macro Avg	0.93	0.93	0.93	6984
Weighted Avg	0.93	0.93	0.93	6984

Table 6.3: Accuracy Report of Bidirectional LSTM for Rab vs. Non Rab data

The image depicts a table containing the Bidirectional LSTM Model's precision, recall, F1 score, support, accuracy, macro average, and weighted average as they pertain to the comparison of data from the Rabindranath era and the present era.Here,Rabindranath Tagore era means 1 and current era means 0. The table presents the model's macro average F1 score of 0.93 and accuracy of 0.93, indicating a satisfactory overall performance. Furthermore, it is important to note that no class imbalance is visible as classes '0' and '1' hold an equivalent number of samples (support value) of 3492. The weighted average F1-score provides an accurate evaluation of the model's performance across all classes by considering the performance of both classes. The weighted average F1 score is 0.93 in this instance as well, suggesting that the model is operating effectively across both divisions. The precision, recall, and f1-score for class 1 are all exceptionally high (approximately 0.93), suggesting that the model demonstrates a high level of accuracy in classifying instances. The performance is also solid for class 0, with precision, recall, and f1-score all hovering around 0.93.
In general, the image demonstrates that the model performs admirably in the classification assignment. Consequently, accurately interpreting the results will be simplified.

Evaluation of Bidirectional LSTM Model:

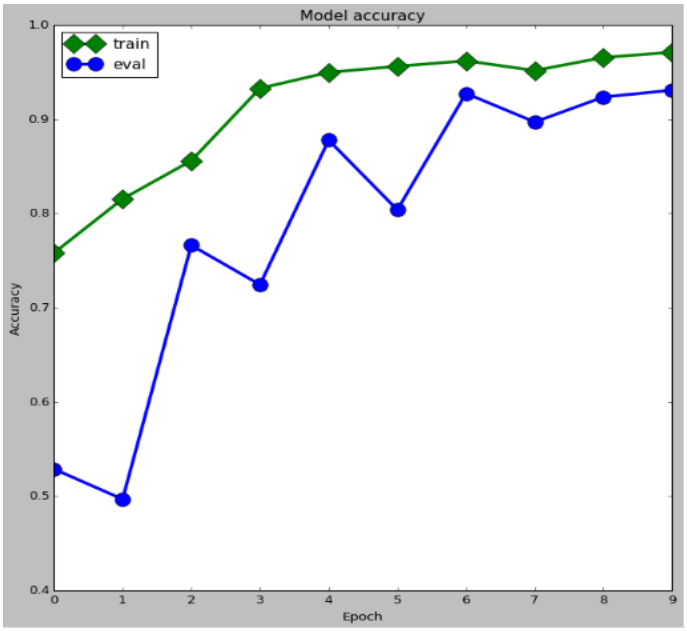


Figure 6.9: Epoch vs Accuracy

Here we can see that the accuracy of the model increases over time, reaching a peak of about 0.92 at epoch 6 and then fluctuates and finally steadily increases to almost about 0.95. This suggests that the performance of the model during evaluation was quite good. This graph also displays that the model is learning and improving over time. The initial increase in accuracy is likely due to the model finding patterns in the data. The blue line shows the accuracy of the model on the validation dataset. It follows a similar trend to the training accuracy, but it is consistently lower. This suggests that the model is overfitting to the training data to some extent. The gap between the blue and green lines is a measure of the model's overfitting. A larger gap indicates that the model is memorizing the training data rather than learning generalizable rules. In this case, the gap is relatively small, which suggests that the model is not overfitting too badly. Overall, this graph suggests that the model is learning well from the training data and is achieving good accuracy on both the training and validation datasets. However, there is some evidence of overfitting, so it might be helpful to try to reduce the gap between the training and validation accuracy.

The below graph is an illustration of model loss over time. The x-axis of the graph represents the number of epochs. The y-axis of the graph represents the model loss, which is a measure of how well the model is able to fit the training data. The graph shows that the model loss for the training dataset decreases sharply up to 2 epochs, and then increases at epoch 3 again and fluctuates until all other epochs are carried out with an overall decrease. This suggests that the model is learning effectively from the training data and reducing its error. The validation dataset follows a similar trend to the training loss, but it is consistently higher. This suggests that the model is starting to overfit to the training data. So, the fluctuation in the graph happened because the model is able to fit the training data perfectly at this point, but it is not able to generalize to new data. Overall, the graph shows that the model is able to learn the training data effectively and reduce its error rate. However, the

graph indicates that the model is not able to improve its performance significantly beyond that point.

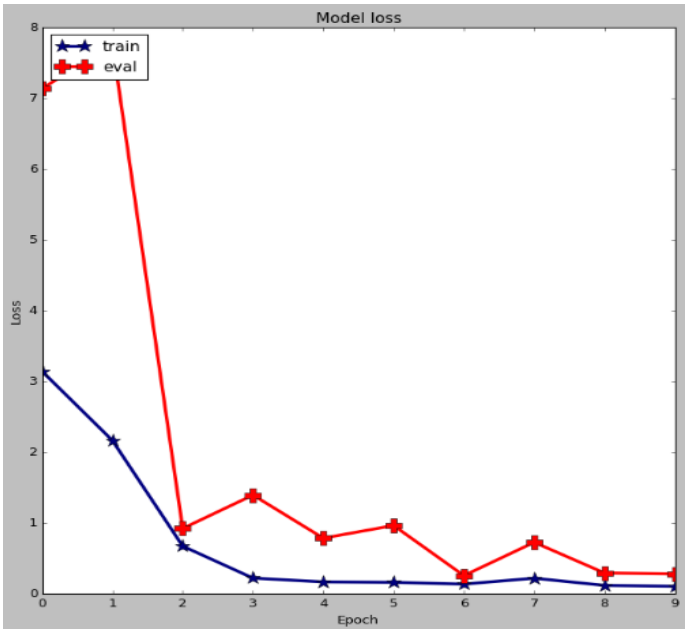


Figure 6.10: Epoch vs Loss

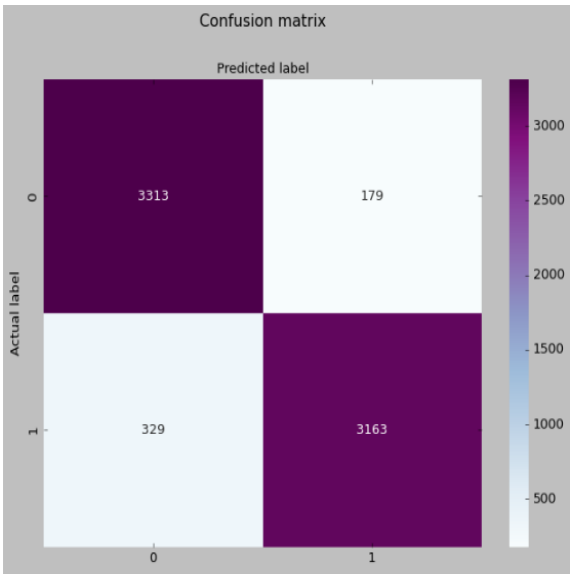


Figure 6.11: Confusion matrix

Here, the confusion matrix helps us understand how well the model is actually working. It has the following values - True Positives (TP): 3313, True Negatives (TN): 3163, False Positives (FP): 179, False Negatives (FN): 329. Based on these values, we can see that the model is overall a good model. It has a high number of true positives and true negatives, which means it is correctly identifying both positive and negative values most of the time. These metrics all suggest that the model is performing well, but there is still room for improvement. In particular, the model could be improved by reducing the number of false positives and false negatives.

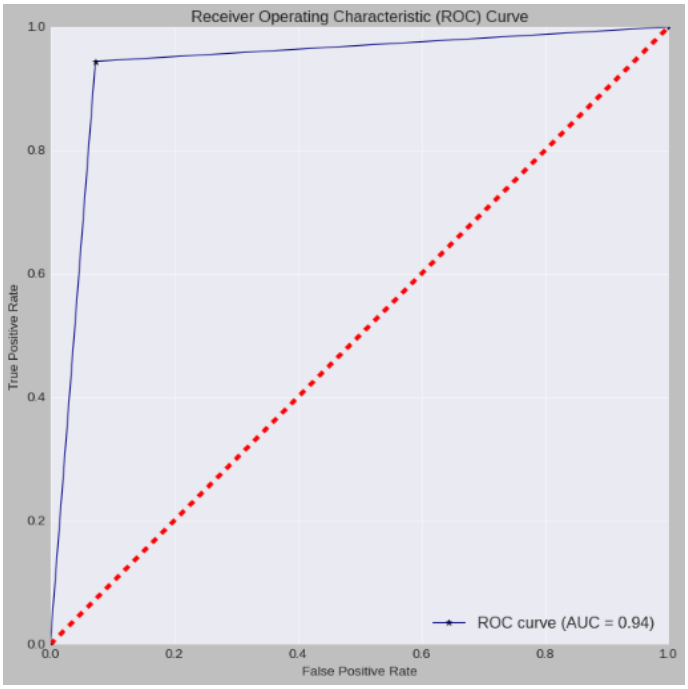


Figure 6.12: ROC Curve

From the above graph we can get a good idea of the overall performance of our model. Here, the y-axis represents the True Positive value rate of our model and the x-axis shows the False Positive rate. Good performance is indicated by the ROC curve positioned in the upper-left corner of the graph, where the True Positive Rate (TPR) is significant and the False Positive Rate (FPR) is minimal. The curve begins off rising steeply, which means that the model is very good at accurately identifying true positives when the threshold is low. As the threshold increases, the curve rises less precipitously, but it remains well above the diagonal line. This means that the model is still able to perform well even when it is more cautious about classifying cases as positive. A uniform curve suggests that the model is making consistent predictions. Then, the AUC (Area Under the Curve) value for this ROC curve is 0.93, which is a very high score. This indicates that the model is very effective at distinguishing between positive and negative cases.

BanglaBERT + Bidirectional LSTM:

Classification Report				
	Precision	Recall	F1-Score	Support
0	0.95	0.99	0.97	3492
1	0.99	0.95	0.97	3492
Accuracy	–	–	0.97	6984
Macro Avg	0.97	0.97	0.97	6984
Weighted Avg	0.97	0.97	0.97	6984

Table 6.4: Classification report of BanglaBERT+LSTM Model (Rob Era vs Current Era)

The table shows the precision, recall, F1 score, support, accuracy, macro average, and weighted average of the Bidirectional LSTM+BanglaBERT Model used to evaluate the Rabindranath era vs. current era data. Here, Rabindranath Tagore era means 1 and the current era means 0.
This is the overall accuracy of the model, calculated as the fraction of all predictions that were correct. In this case, the accuracy is 97%, which is very good. This is the average of the precision, recall, and f1-score across both classes. It gives an overall sense of how well the model is performing on both classes combined. In this case, the macro average is also 97%, which is again very good. This is the average of the precision, recall, and f1-score, weighted by the support of each class. It gives more weight to the performance of the larger class. In this case, the weighted average is also 97%. Furthermore, there is no class imbalance present, as both the classes '0' and '1' have the same support value (number of samples) which is 3492.
Overall, the classification report shows that the model is performing very well on this binary classification task. It has high accuracy, precision, recall, and f1-score on both classes.

Evaluation of BanglaBERT + Bidirectional LSTM Model:

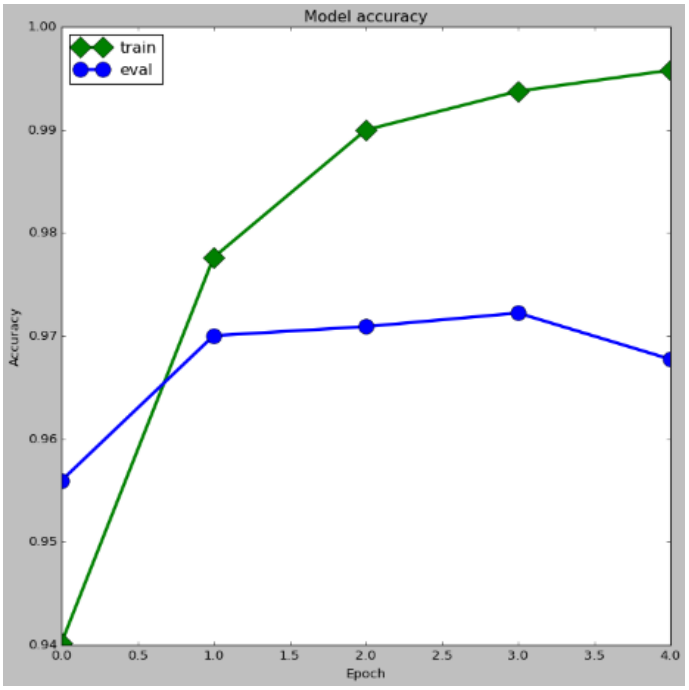


Figure 6.13: Accuracy vs. Epoch

The graph shows both the training accuracy (blue line) and the validation accuracy (orange line). The training accuracy is generally higher than the validation accuracy, which is expected.
Both training and evaluation accuracy start relatively low and increase rapidly during the first few epochs. Furthermore, the lines converge around epoch 0.7, indicating that the model's performance on both training and evaluation data is stabilizing. And then, after epoch 1.0, the training accuracy continues to increase slightly, while the evaluation accuracy plateaus or even decreases slightly. This suggests potential

overfitting, where the model is becoming too specialized to the training data and not generalizing well to unseen data. Overall, the model performs well.

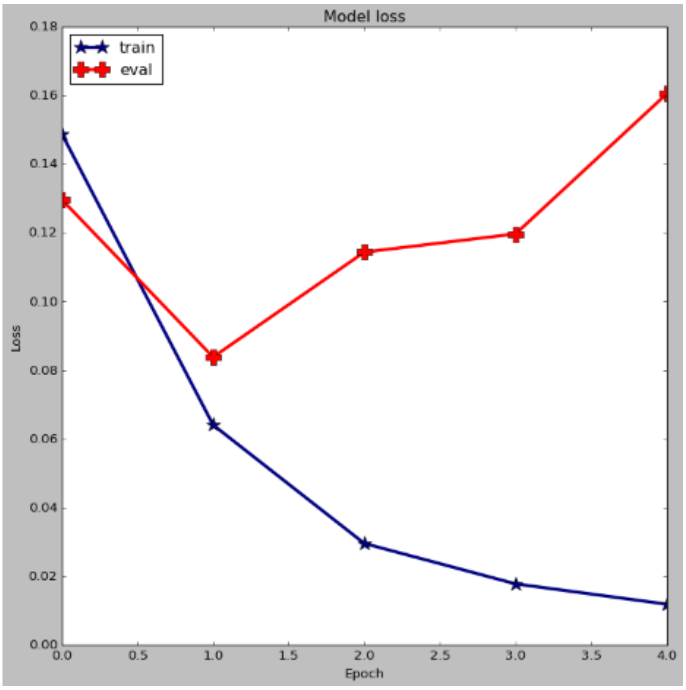


Figure 6.14: Loss vs Epoch (2)

The training loss typically decreases as the model is trained, as it learns to better fit the training data. However, the evaluation loss may not always decrease as the model is trained.
In the graph above, the training loss appears to decrease steadily over time, while the evaluation loss remains relatively constant. This suggests that the model is learning to fit the training data well without overfitting. This is a good sign, as it means that the model is likely to perform well on new data. Overall, the graph you sent me suggests that the model is training well and is not overfitting the training data.

The confusion matrix of the next page shows the number of data points that were correctly and incorrectly classified by the model. In this specific confusion matrix, the rows represent the actual labels of the data points, while the columns represent the predicted labels. The diagonal cells show the number of data points that were correctly classified. For example, in the top left cell, we can see that 3455 data points were the True Positive values. The off-diagonal cells show the number of data points that were incorrectly classified. For example, in the cell in the second row and first column, we can see that 186 data points were false negative values. We can see the overall performance of the model from the confusion matrix and also calculate its accuracy, precision, recall and f1 score from it.

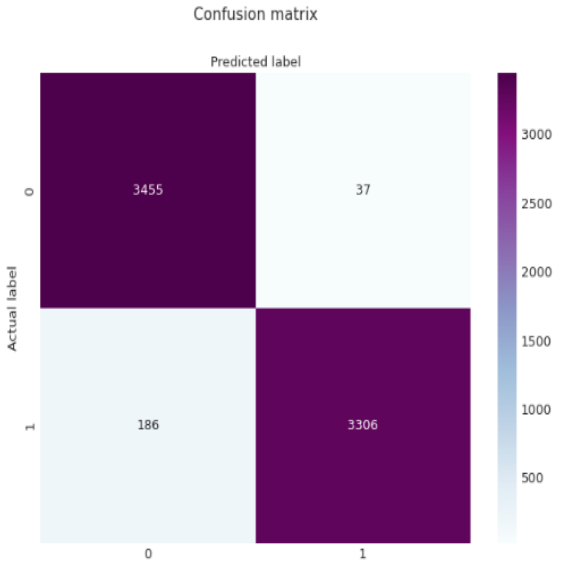


Figure 6.15: Confusion Matrix

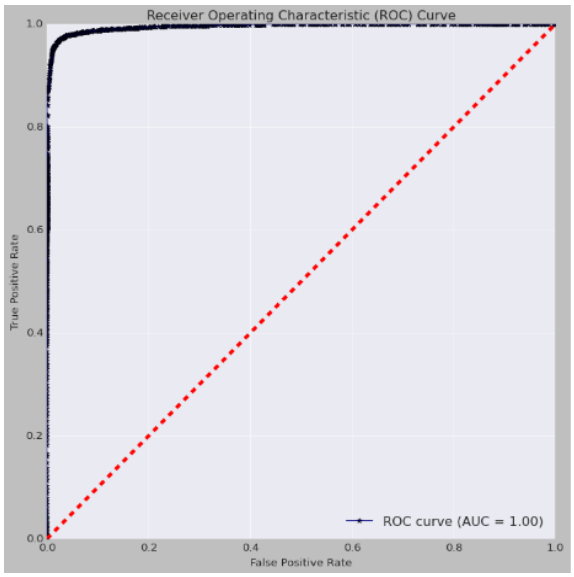


Figure 6.16: ROC Curve

The ROC curve above is a perfect ROC curve. This means that the model is able to perfectly discriminate between positive and negative cases. This is often not the case in real-world applications, where models typically have to make trade-offs between TPR and FPR.
The ROC curve is very close to the top-left corner of the graph, which is an indication of good performance. The curve starts off rising steeply, which means that the model is very good at correctly identifying true positives. A smooth curve suggests that the model is making consistent predictions. Then, the AUC (Area Under the Curve) value for this ROC curve is 1.00, which is an unrealistically good score.

6.1.3 Experimental findings from Rabindranath Era vs. Multi Author Era data

Bidirectional LSTM:

Classification Report				
	Precision	Recall	F1-Score	Support
0	0.92	0.91	0.92	5987
1	0.90	0.86	0.88	6097
2	0.86	0.90	0.88	5948
Accuracy	–	–	0.89	18032
Macro Avg	0.89	0.89	0.89	18032
Weighted Avg	0.89	0.89	0.89	18032

Table 6.5: Classification Report of Bidirectional LSTM for Rabindranath vs Multi Author data

In the above table, we can the precision, recall, f1 score, accuracy, macro average and weighted average of the Bidirectional LSTM Model report. Here, it has been used to evaluate the Rabindranath vs Multi Author data. Here, Rabindranath Tagore means 1, Sarat Chandra Chattopadhyay means 2 and Bankim Chandra means 0.

After critically analyzing the data of the report, it is visible that the model has a decent accuracy rate. The accuracy of the entire model is 0.89. Additionally, there is a visible class imbalance present in this report, as the classes '0' , '1' & '2' have different support values. For class 0, it's 5987, for class 1, it's 6097, and for class 2, it's 5948. The recall is 0.91 for class 0, 0.88 for class 1, and 0.88 for class 2. Also, the F1-scores are 0.92 for class 0, 0.88 for class 1, and 0.88 for class 2. This also shows the variance in values of the balance between precision and recall. The weighted average F1-score takes into account the performance of both classes and gives an accurate assessment of the model's performance on all classes. In this case, the weighted average F1 score is also 0.89, indicating that the model is performing well on both classes. As for an overview of the report, the values of precision, recall & F1-score of all 3 classes are above 0.89, which are very high. It indicates that the model is good at correctly identifying cases that belong to this class.

Overall, we can see that the image reflects the model as a good performer as it can accurately do the classification task. Thus, it will be easier for it to interpret the results accordingly.

Evaluation of Bidirectional LSTM Model:

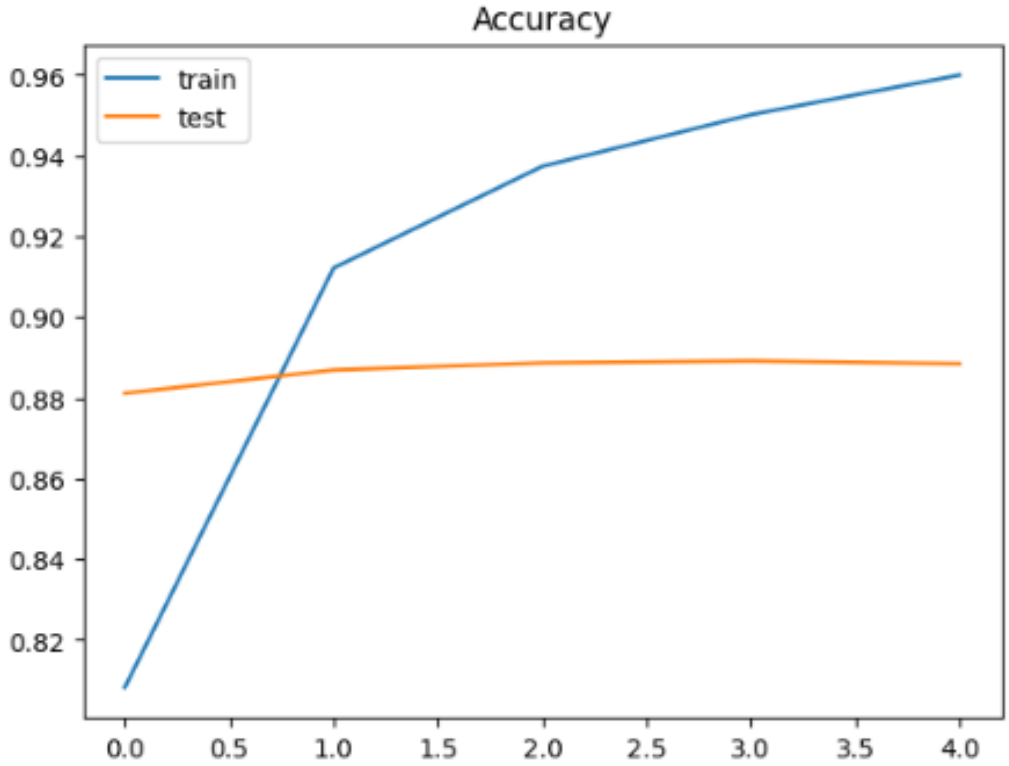


Figure 6.17: Epoch vs Accuracy of Bidirectional LSTM for Rabindranath vs Multi Author data

In the given graph, we can see the accuracy of the model remains relatively flat and does not improve much as the training progresses. With a slight increase from over 88% at Epoch 0.0, we can see a peak of about 89% at Epoch 2.0. This reflects that the performance of the model during the evaluation period was relatively stable. This graph also shows that the model is having a slight improvement and learning more by time. The slight increase in accuracy till Epoch 2.0 shows that the model is performing well, steadily improving, and is able to achieve high accuracy on the test data.

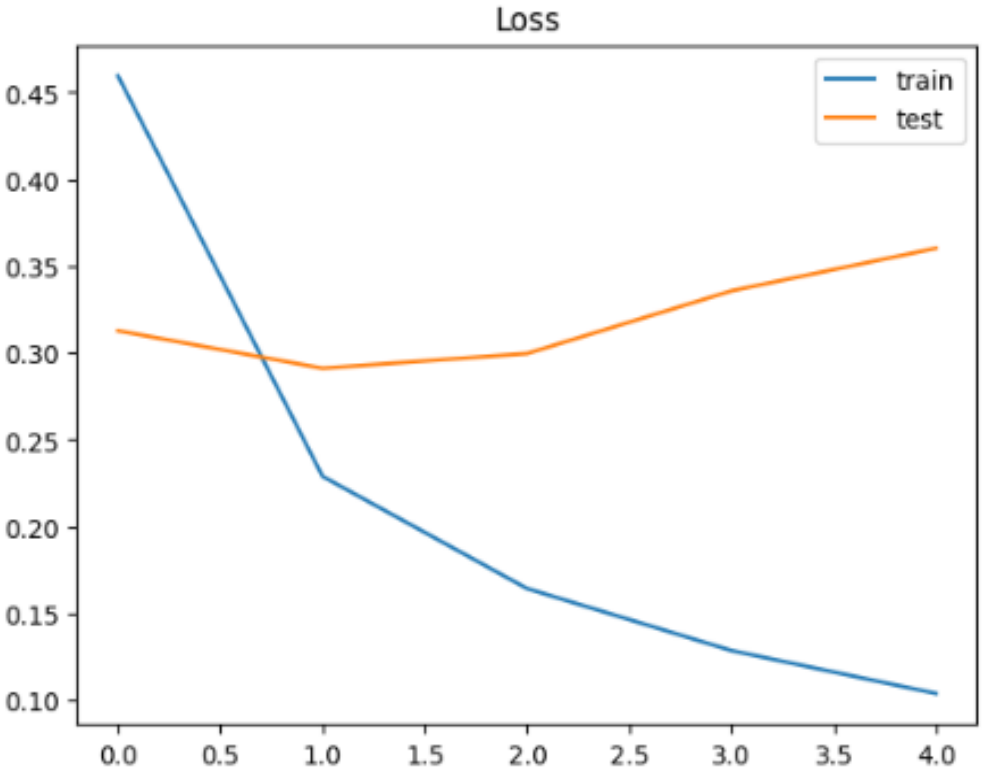


Figure 6.18: Epoch vs Loss of Bidirectional LSTM for Rabindranath vs Multi data

The first graph on the next page shows that the loss of the model initially decreases slightly, and right after 0.31 at Epoch 1.0, the graph starts to slightly improve to nearly 0.34 at Epoch 2.0, suggesting that the model's performance on the test data is not improving gradually as training continues.

This graph shows that the model is developing itself and learning. The improvement is visible from the graph. The first decline in loss is likely due to the model finding patterns in the data and the subsequent increase in loss may be due to the model overfitting to the table dataset.
Therefore, it is visible that the model will be able to achieve low loss on the test data and the model is performing well.

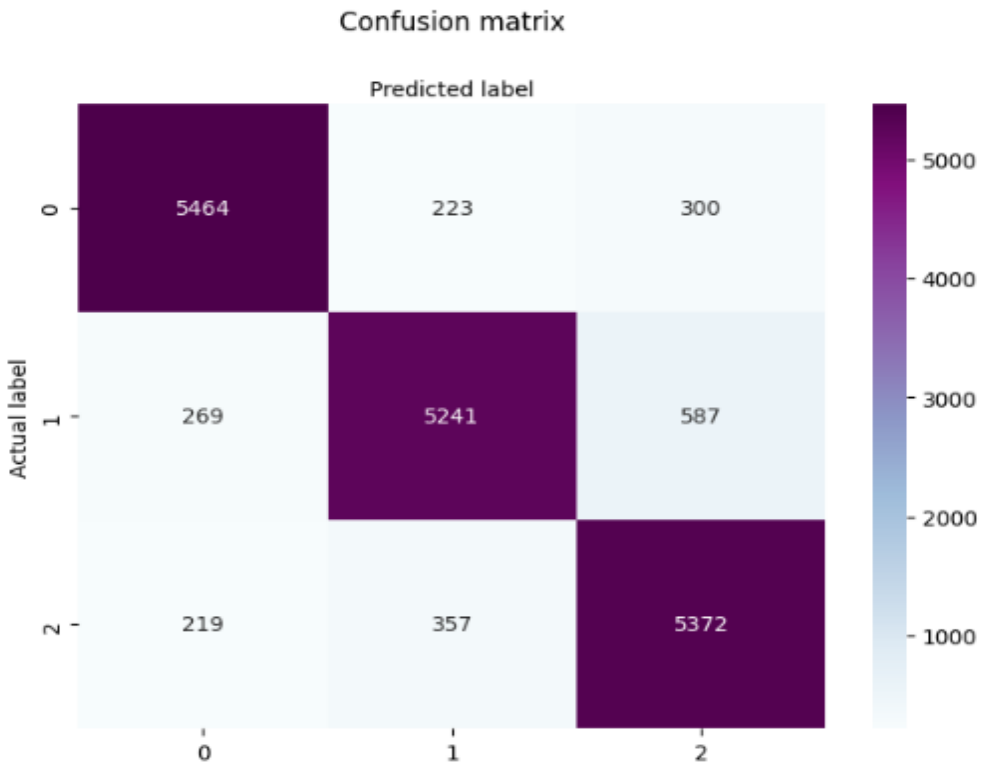


Figure 6.19: Confusion Matrix of Bidirectional LSTM for Rabindranath vs Multi Author data.

The image shows the confusion matrix, which helps us understand how well the model is actually working. As we have 3 classes, the matrix is a 3x3 grid, indicating it evaluates a model classifying instances into 3 distinct classes (labeled 0, 1, and 2). For example, the cell in the second row and third column shows the value ”5241”. This means that there were 5241 instances where the model actually belonged to class 2, but the model predicted it as class 1. Each cell contains the support values for each class. It shows that the model is performing well overall. **BanglaBERT + Bidirectional LSTM:**

Classification Report				
	Precision	Recall	F1-Score	Support
0.0	0.92	0.94	0.93	3996
1.0	0.91	0.90	0.91	3993
2.0	0.91	0.91	0.91	3993
Accuracy	–	–	0.92	11982
Macro Avg	0.92	0.92	0.92	11982
Weighted Avg	0.92	0.92	0.92	11982

Table 6.6: Accuracy Report of Bidirectional LSTM for Rabindranath vs Multi Author data

The table given above is the report which summarizes the model's execution level on a test dataset, breaking down it's prediction for each class into several metrics such

as - precision, recall, F1 score, support, accuracy, macro average and weighted average for BanglaBERT + Bidirectional LSTM Model. It will be used to evaluate the given Rabindranath vs. Multi Author data in the report.Here, Rabindranath Tagore means 1, Sarat Chandra Chattopadhyay means 2 and Bankim Chandra means 0.

From the table, we can see after evaluating the values and results that the model has an overall good performance. The overall accuracy of the model is 0.92. This means that the model correctly classified 92% of the examples in the test data. This indicates the overall proportion of the correct predictions made by the model across all the 3 classes. It also shows that the macro average value is 0.92. And if we can see, there is a bare minimum difference in support values between class 0 and class 1 & 2, whereas the support value of both class 1 & 2 is the same. The support for class 0 is 3996, the support for class 1 is 3993, and the support for class 2 is 3993. Now evaluating the precision, recall and F1-score values of class 1, 2 & 3, we can see all of them have pretty high values. The precision value for class 0 is 0.92, the precision value for class 1 is 0.91, and the precision value for class 2 is 0.91. Similarly, the recall and f1-scores of these 3 classes are pretty high.

This classification report suggests that the model is performing well overall on this dataset. It is able to accurately identify both positive and negative cases with high precision and recall.

Evaluation of BanglaBERT vs Bidirectional LSTM:

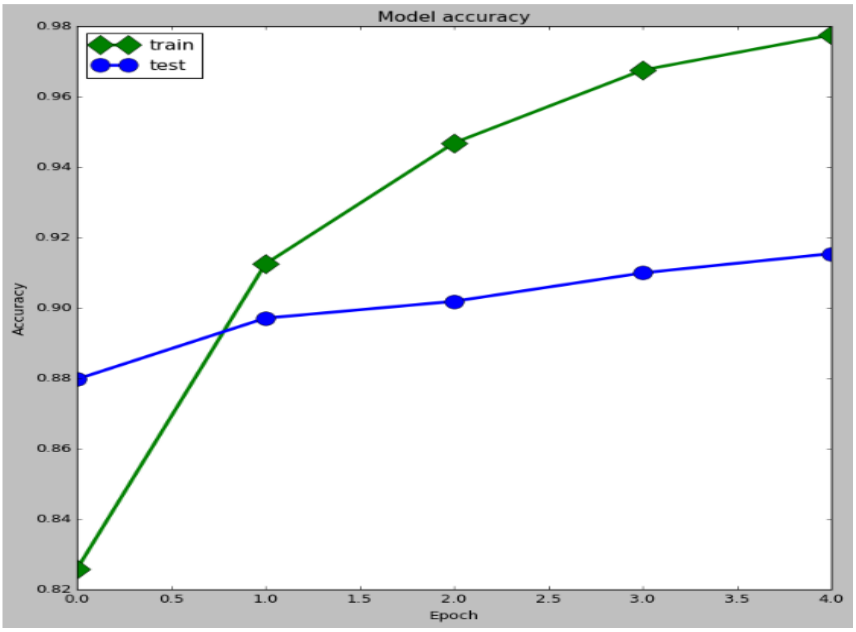


Figure 6.20: Epoch vs Accuracy BanglaBERT vs Bidirectional LSTM.

In the image given below is the accuracy graph of the model which, according to the lines, is increasing over time. The model achieves its highest accuracy around epoch 1.0, with a test accuracy of approximately 0.90. After epoch 1.0, the accuracy curves start to plateau. But then the accuracy value stands at 0.91 (approx.) at Epoc 2.0.

Additionally, this graph shows how the model is developing and learning over time. The model's ability to identify patterns in the data is probably what led to the

initial accuracy boost. The accuracy increase reduces after epoch 1.0 as the gradient becomes less inclined. The model might have overfitted to the training set as a result of this. Overall, the graph indicates that the model is operating effectively and achieving high accuracy on the test data.

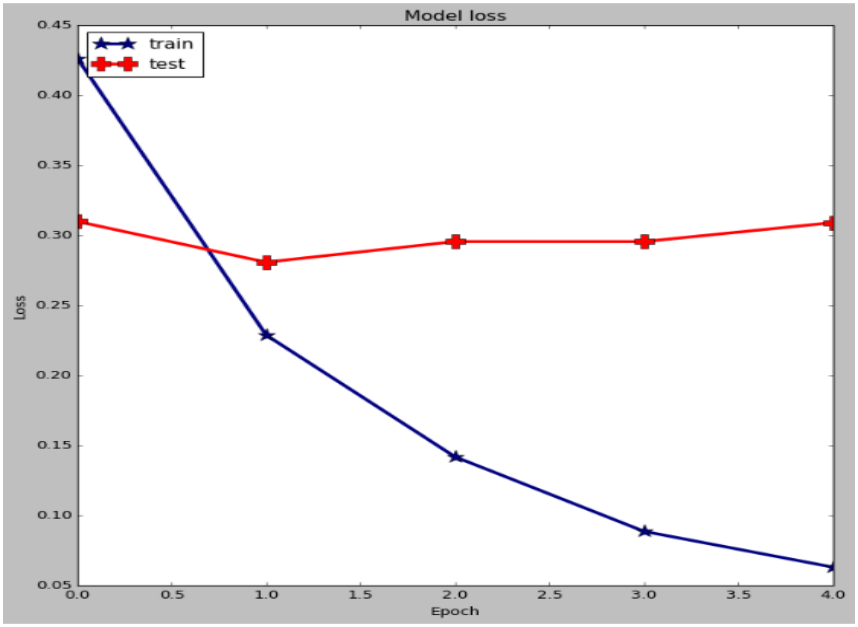


Figure 6.21: Epoch vs Loss of BanglaBERT vs Bidirectional LSTM.

In the image given above is the epoch vs loss graph of the model which, according to the lines, is decreasing over time. The graph shows that the model loss decreases up to 0.26 at Epoch 1.0 for test data, and then plateaus up to Epoch 2.0. It stays at 0.27 on Epoch 2.0. This indicates that the model is able to learn the training data effectively and reduce its error rate.
The plateau in the graph indicates that the model has reached its maximum ability to learn from the training data. The reason behind this is, the lack in ability to generalize to new data even though the model can fit the training data.

This graph illustrates how the model is developing and learning over time. The initial decrease in loss is likely due to the model finding patterns in the data and the subsequent increase in loss may be due to the model overfitting to the training dataset. In general, the graph indicates that the model is operating effectively and achieving little loss on the test data.

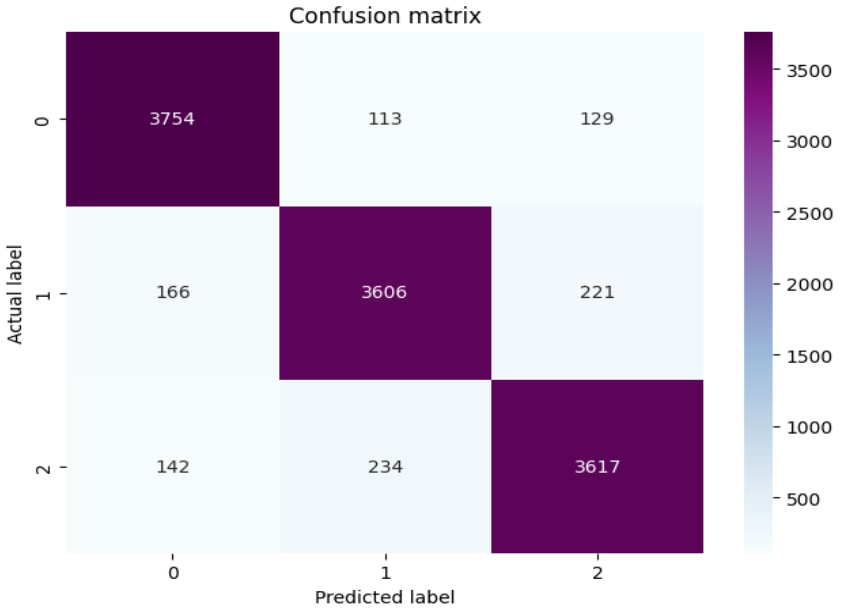


Figure 6.22: Confusion Matrix of BanglaBERT vs Bidirectional LSTM

Above image is the confusion matrix, which helps us understand how well the model is actually working. In this case, there are three classes (0, 1, and 2), which suggests that the algorithm was solving a three-class classification problem. That is why it is seen as a 3x3 matrix. The values in these cells are the support values which are 3754 for class 0, 3606 for class 1, and 3617 for class 2, indicating that the classifier most accurately predicted class 2 and least accurately predicted class 0. So, the overall performance of the model is quite good according to this confusion matrix.

6.1.4 Experimental findings from Adversarial Data:

Classification Report				
	Precision	Recall	F1-Score	Support
0	0.86	0.84	0.85	3271
1	0.84	0.86	0.85	3271
Accuracy	–	–	0.85	6542
Macro Avg	0.85	0.85	0.85	6542
Weighted Avg	0.85	0.85	0.85	6542

Table 6.7: Classification Report of Bidirectional LSTM for Adversarial Data.

The table is a classification report for Adversarial Data which contains some metrics that allows us to determine the overall performance of our Bidirectional LSTM model. It has two rows, representing the two classes in the classification task. In this case, the classes are labeled ”0” and ”1”.Here, Rabindranth Tagore texts denotes 1 and model generated texts denotes 0.

Overall, the image shows that the model has good performance on the classification task. Thus, it will be easier to interpret the results accordingly.

Evaluation of Bidirectional LSTM:

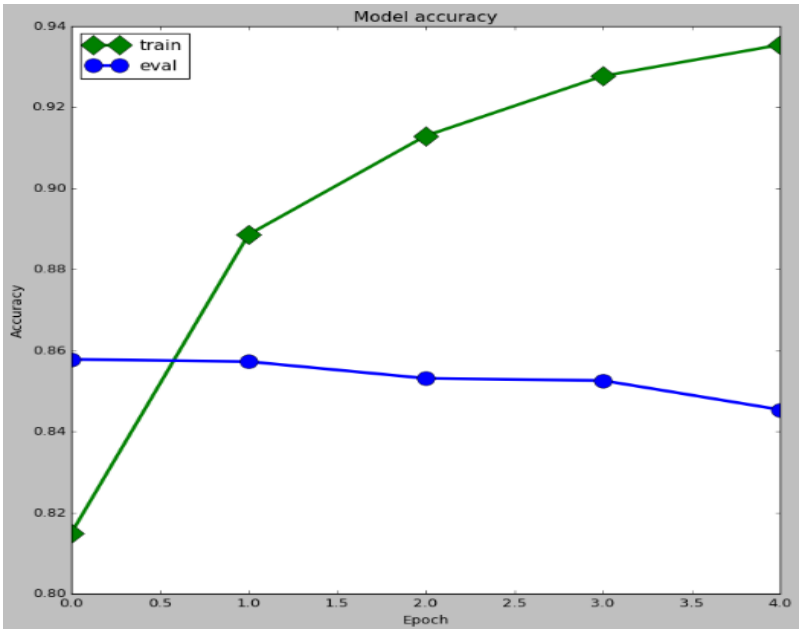


Figure 6.23: Accuracy vs Epoch of Bidirectional LSTM

This graph represents epoch in its x-axis and accuracy of the model in its y-axis. Furthermore, blue line is training dataset accuracy and green line is test dataset accuracy.

As it is evident, the accuracy of the test dataset is lower than that of the training dataset. This happens because overfitting might have occured. This actually means that the training dataset learned the pattern of the data too well and cannot generalize to new data, or in other words, test data anymore. This phenomenon of overfitting might have happened because there is not enough training data. If there is not enough training data, then the model will not be able to learn the underlying patterns in the data.

In the first image of next page, we can see the Epoch vs loss graph where it is visible that the accuracy of the test data in the graph slightly increased to its peak value of almost 0.4 at Epoch 3.0. The training process, in blue, starts at an average loss of around 0.36 and appears to steadily decrease to a little under 0.15. The eval process, in orange, starts at a higher average loss, near 0.3, and increases then. By the end of the graph, the test process has an average loss around 0.4.
Overall, the graph suggests that the training process is performing better than the test process. This could be due to overfitting, where the model is too closely fitted to the training data and doesn't generalize well to new data.

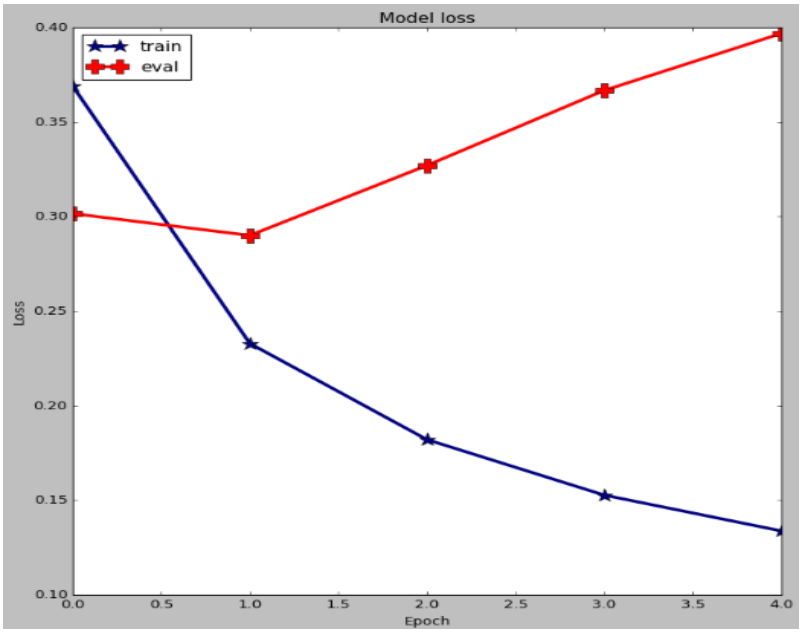


Figure 6.24: Epoch vs Loss of Bidirectional LSTM

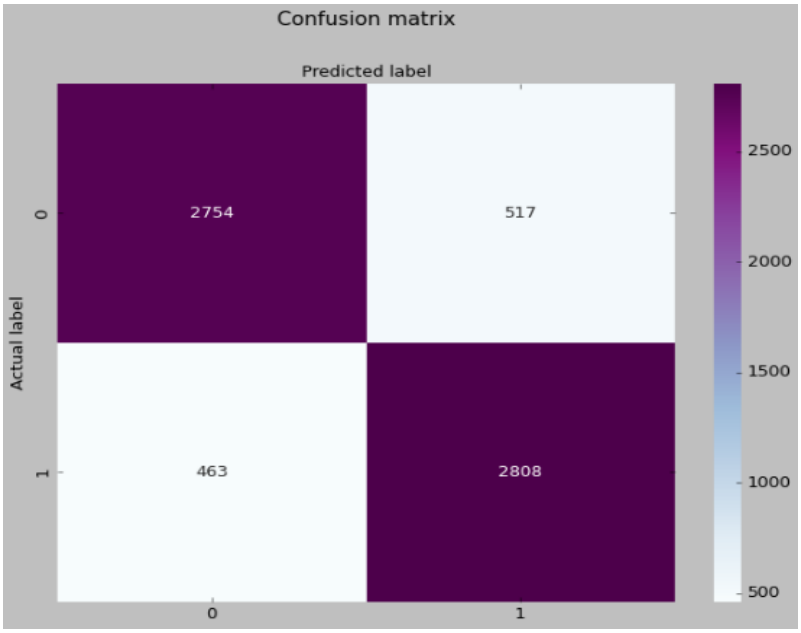


Figure 6.25: Confusion matrix of Bidirectional LSTM

The rows of the matrix represent the actual labels of the data, and the columns represent the predicted labels of the model.
Here, the cell in the first row and first column shows that there were 2754 data points that were actually labeled as class 0 and were also predicted as class 0 by the model. And the cell in the second row and first column shows that there were 463 data points that were actually labeled as class 1 but were predicted as class 0 by the model. The model seems to be performing well on class 1, with a high number of correct predictions 2808. The model is less accurate on class 0, with a higher number of incorrect predictions 517.
These metrics all suggest that the model is performing well, but there is still room for improvement. In particular, the model could be improved by reducing the number of false positives and false negatives.

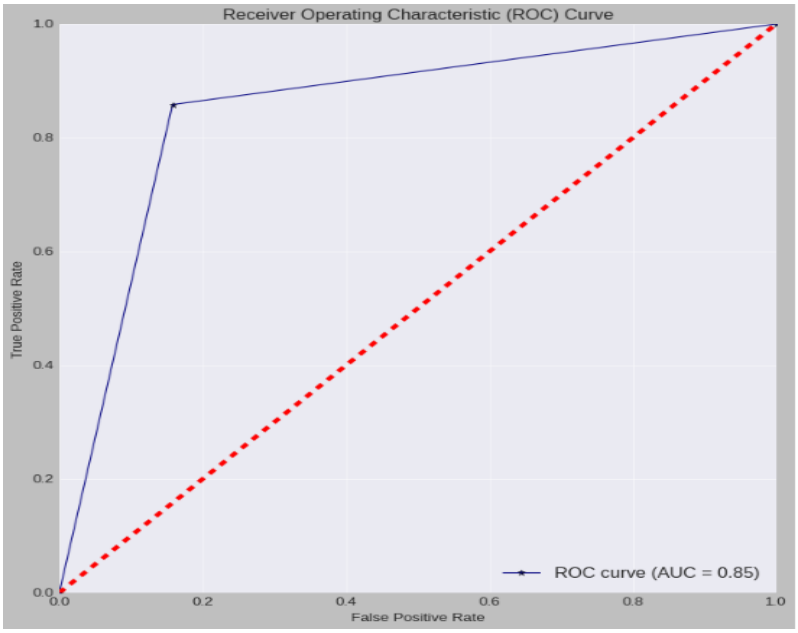


Figure 6.26: ROC Curve for Bidirectional LSTM

This is a ROC curve and we can see that the curve is close to the top-left corner of the graph. This is a good sign. Moreover, the AUC (Area Under the Curve) is 0.86 which is also very good. This means that the model is able to distinguish between the two classes very well, even at low levels of certainty and the overall performance of the model is very good. Furthermore, the curve is smooth and does not have any sharp drops, Which ultimately means that the curve is not much sensitive to minor changes and is much stable. Interpretations of the data is made upon this and all predictions are rather consistent to each other.

BanglaBERT + Bidirectional LSTM:

Classification Report				
	Precision	Recall	F1-Score	Support
0	0.91	0.94	0.92	3285
1	0.94	0.90	0.92	3285
Accuracy	–	–	0.92	6543
Macro Avg	0.92	0.92	0.92	6543
Weighted Avg	0.92	0.92	0.92	6543

Table 6.8: Classification Report

The image shows a table of precision, recall, F1 score, support, accuracy, macro average, and weighted average of the Bidirectional LSTM+BanglaBERT Model used to evaluate the Adversarial data.

In the above table, we can the precision, recall, f1 score, accuracy, macro average and weighted average of the Bidirectional LSTM Model report. Here, it has been used to evaluate the Rabindranath vs Multi Author data.Here, Rabindranth Tagore texts denotes 1 and model generated texts denotes 0.

After critically analyzing the data of the report, it is visible that the model has a decent accuracy rate. The accuracy of the entire model is 0.92. Additionally, there is a visible class imbalance present in this report, as the classes '0' & '1' have different support values. For class 0, it's 3285, for class 1, it's 3285. Furthermore we can see that the weighted average F1 score is also 0.92, indicating that the model is performing well on both classes. Overall, we can see that the image reflects the model as a good performer as it can accurately do the classification task. Thus, it will be easier for it to interpret the results accordingly.

Evaluation of BanglaBERT + Bidirectional LSTM:

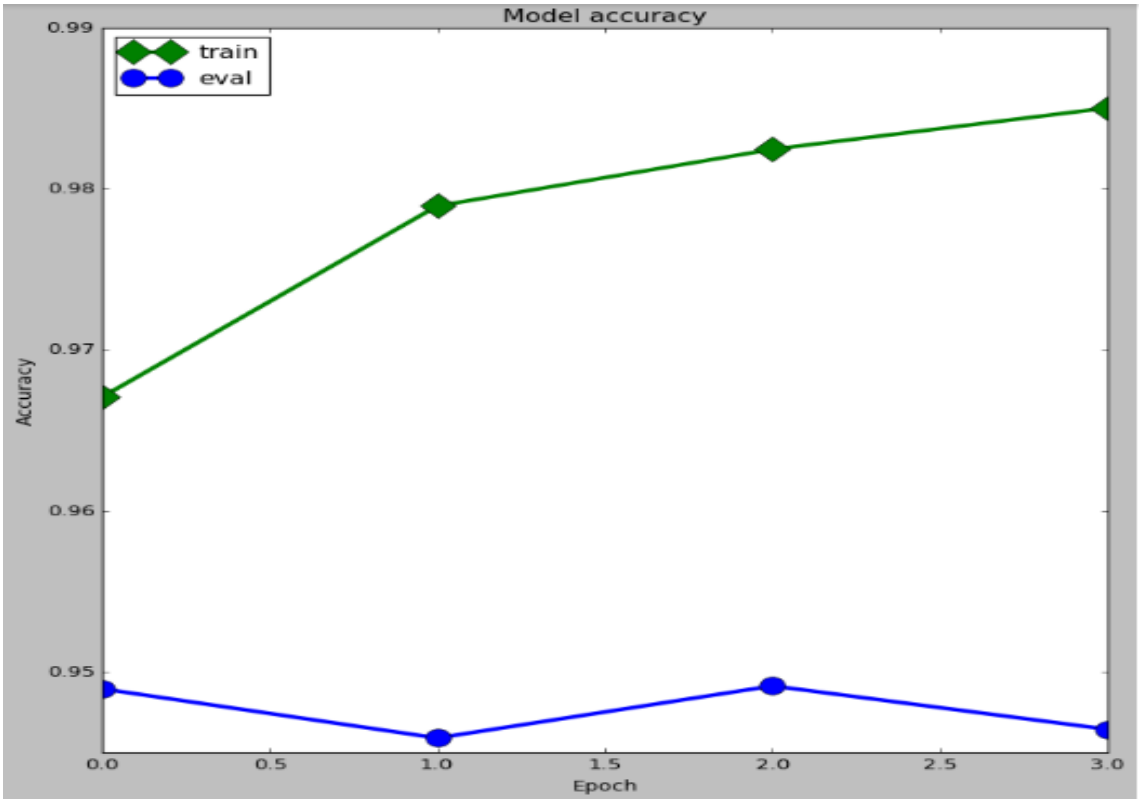


Figure 6.27: Accuracy vs Epoch

The graph shows the accuracy of a language model over time. The training accuracy starts out at around 0.96 and increases to about 0.99 over the course of training. This suggests that the model is learning to fit the training data well. However, the validation accuracy starts out at around 0.94 and then decreases to about 0.92 over the course of training. This suggests that the model is overfitting the training data. Overfitting occurs when a model memorizes the training data too well and is unable to generalize to new data.

The graph of the next page shows that the training loss (blue line) is decreasing over time, which means that the model is getting better at fitting the training data. However, the validation loss (red line) is increasing over time, which means that the model is not generalizing well to unseen data. This is a sign that the model is overfitting. Overfitting occurs when a machine learning model learns the training data too well, but is unable to perform well on new data.

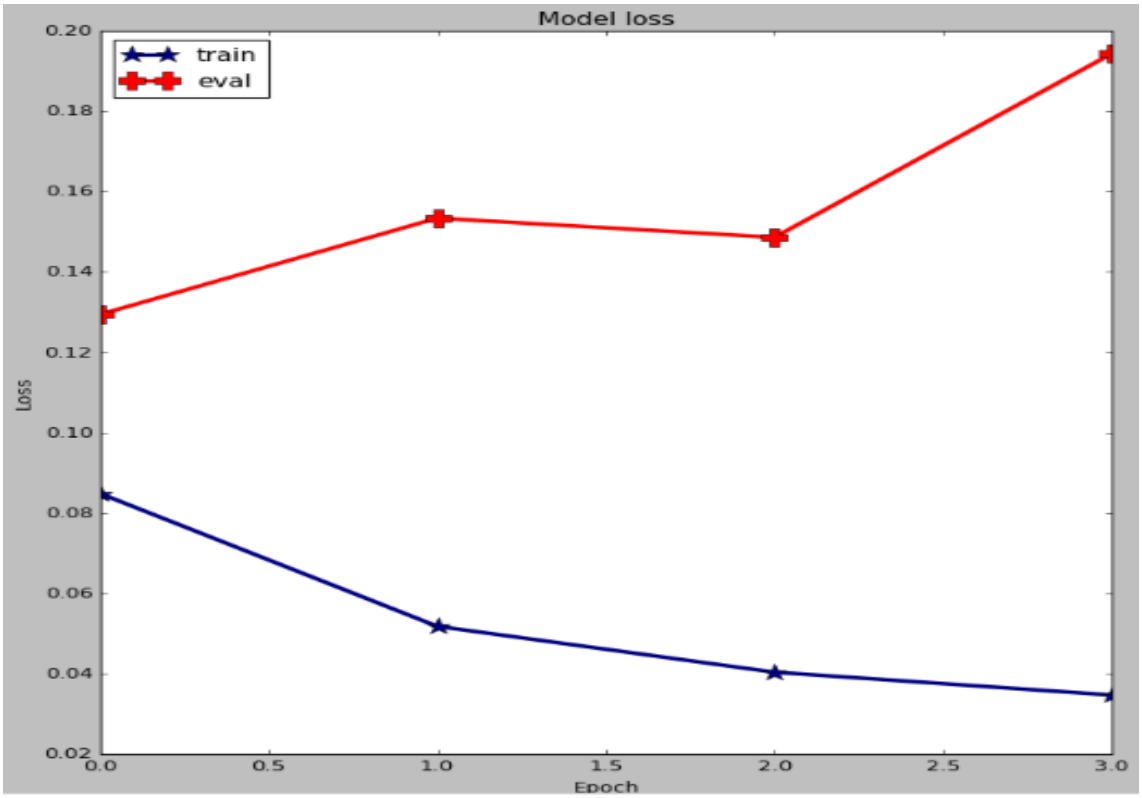


Figure 6.28: Loss vs Epoch

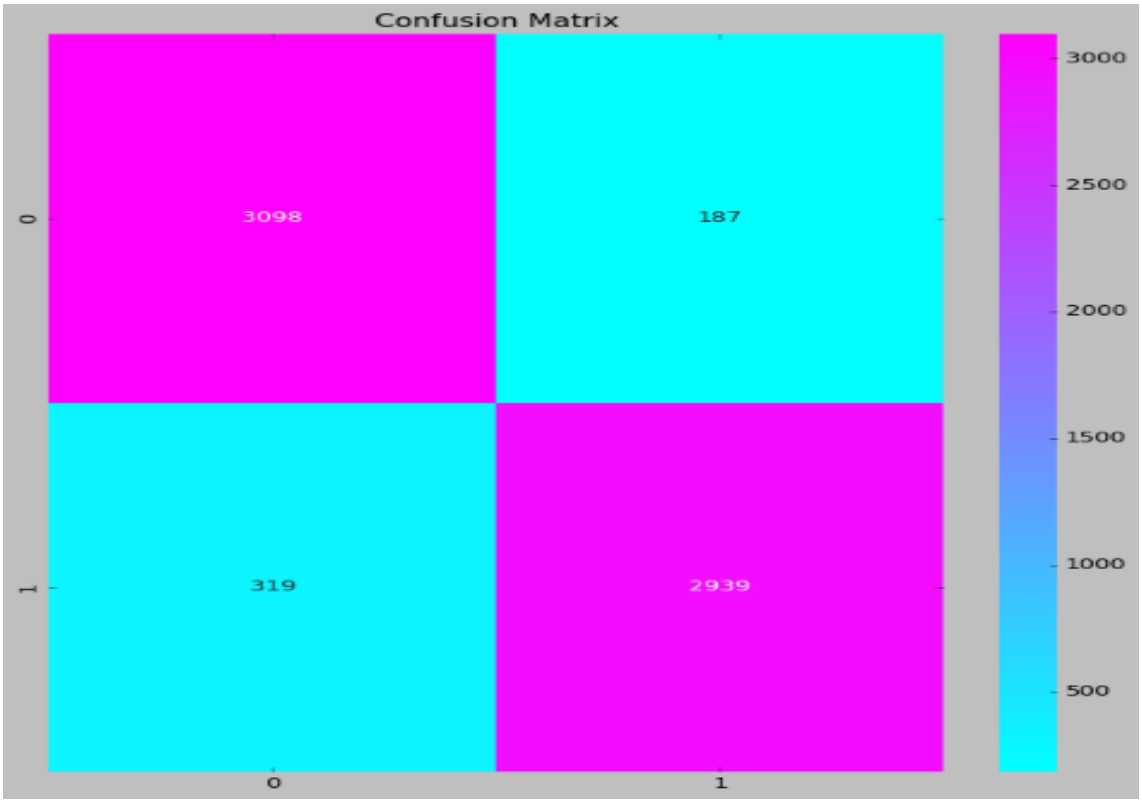


Figure 6.29: Confusion Matrix

The rows represent the actual classes, and the columns represent the predicted classes. The numbers in the cells represent the number of data points that were classified into each combination of actual and predicted class.
The model correctly classified 3098 Non-Rabindranath texts datas and incorrectly classified 187. Overall, the model is performing well with an accuracy of over 92%. However, the model classified 2939 Rabindranath texts correctly but 319 texts incorrectly.

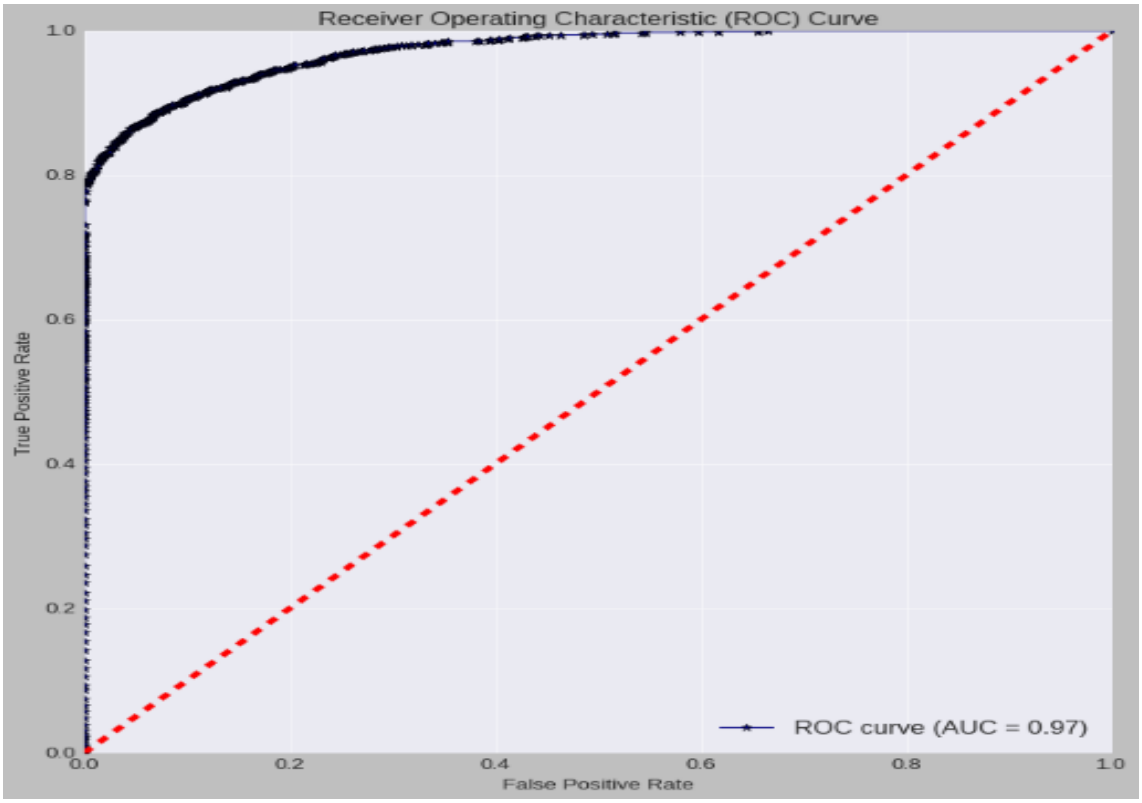


Figure 6.30: ROC Curve

The curve is close to the top-left corner of the graph, which is a good sign, and the AUC (Area Under the Curve) is .97, which is also very good. This means that the model is able to distinguish between the two classes very well, even at low levels of certainty and the overall performance of the model is very good. But this is not realistically possible. The curve is smooth and does not have any sharp drops, Which ultimately means that the curve is not much sensitive to minor changes, and is much stable. Interpretations of the data is made upon this and all predictions are rather consistent to each other.

6.2 Text Generation

After analyzing the writing style of Rabindranath, we devoted ourselves to the task of generating texts in Rabindranath's style. For the purpose we finetuned Csebuetnlp's Bangla T5 and google's mt5-small. And from the training vs validation loss curve from **Figure 6.31** and **Figure 6.32** that we got while finetuning BanglaT5 and mt5 small respectively. In both the graphs we can see that initially the training loss decreases rapidly and similarly the validation loss too drops but not as rapidly as the training curve. Later both the training and validation losses decrease steadily throughout the training. This shows that the models are effectively learning from the training data and generalizing well to new, unseen data.

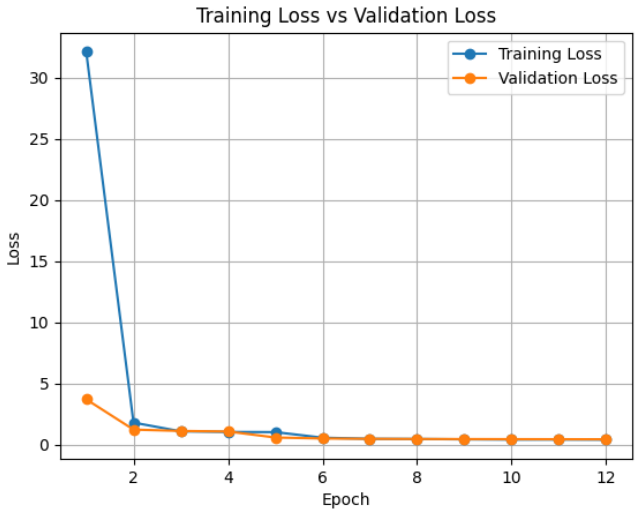


Figure 6.31: Training Vs Validation loss (*BanglaT5*)

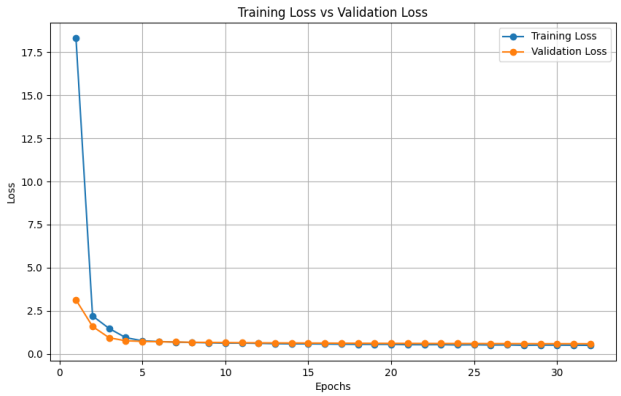


Figure 6.32: Training Vs Validation loss (*mT5-small*)

We used the BLEU(Bilingual Evaluation Understudy) score to evaluate the quality of our generated texts and below we can find the distribution of the BLEU score of different ranges for our test dataset of 3000 sentences.

Here in the **Table 6.9** we can see that for both the models, the maximum of the

Table 6.9: Translation Quality Evaluation

BLEU Score	Translation Quality	Text Amount (BanglaT5)	Text Amount (mT5)
< 10	Almost useless	0.83%	0.53%
$10 - 20$	Hard to get the gist	3.1%	4.33%
$20 - 30$	Clear but significant grammatical errors	6.43%	9.80%
$30 - 40$	Understandable to good translations	12.9%	18.8%
$40 - 50$	High-quality translations	21.7%	24.9%
$50 - 60$	Very high quality, and fluent translations	21.53%	22.53%
$60 >$	Quality often better than human	33.5%	19.1%

Table 6.10: Model Performance Comparison

Models	Avg BLEU Score (Train)	Avg BLEU Score (Test)	Avg BLEU Score (Validation)	Time to Train
BanglaT5	50.54	52.49	54.48	∼4.5 hours
mT5-small	52.19	46.88	44.57	∼6 hours

generated texts scored above 40 but the BanglaT5 model performed substantially better than the mT5-small model as it has more than 76.73% of the generated texts scoring more than 40 whereas for mT5-small it was 66.53%. However, the sentences that scored pretty less i.e. less than 30 were actually too large input sentences for which the model couldn't properly transfer the style of Rabindranath.

Overall Comparison: In the **Table 6.10**, if we compare, it becomes quite visible that BanglaT5 performed better in terms of better BLEU Score and consumption of computational resources and time.

A demonstration of our model generated texts using finetuned BanglaT5 and mT5-small has been shown in **Table 6.11**.

Table 6.11: Translation Comparison

Input Text	**Reference Text (Rabindranath)**	**BanglaT5 Generated Text**	**mT5-small Generated Text**
গোরার কোন আইনজীবী ছিল না।	গোরার উকিল কেহ ছিল না।	গোরার উকিল ছিল না।	গোরার কোন আইনজীবী ছিল না।
হারান যে-উদ্দেশ্যে এই গল্প বলেছিল, তা পুরোপুরি ব্যর্থ হয়ে গিয়েছিল।	হারান যে উদ্দেশ্যে এই গল্পটা বলিলেন তাহা সম্পূর্ণ ব্যর্থ হইয়া গেল।	হারান যে-উদ্দিষ্টে এই গল্প বলিয়াছিল, তাহা একেবারে ব্যর্থ হইয়া গেল।	হারান যে-উদ্দেশ্যে এই গল্পটা কহিল, তাহা সম্পূর্ণ ব্যর্থ হইয়া গেল।
সূর্য যখন মাথার ওপর থেকে পশ্চিমে হেলে যায়, তখন একটা গাড়ি তার সামনে এসে থামে।	সূর্য মাথার উপর হইতে পশ্চিমের দিকে যখন হেলিয়াছে তখন একটা গাড়ি ঠিক তাহার সম্মুখে আসিয়া থামিল।	সূর্য যখন মাথার উপর থেকে পশ্চিমে ঝুঁকছে, এমন সময় একটা গাড়ি তার সামনে এসে দাঁড়াল।	সূর্য যখন মাথার উপর হইতে পশ্চিমে হেলিয়া গেল, তখন একটা গাড়ি তাহার সামনে আসিয়া থামিয়া উঠিল।
সে দ্রুত গাড়িতে উঠে পড়ে।	সে তাড়াতাড়ি গাড়িতে উঠিয়া পড়িল।	সে তাড়াতাড়ি গাড়িতে উঠে পড়ল।	তাড়াতাড়ি গাড়িতে উঠিয়া পড়িল।
আমি আর পছন্দ করি না, স্যার, এক সময় আমি আমার গলায় দড়ি দিয়ে মরতে চাই!	আর ভালো লাগে না মশায়, এক-এক সময় ইচ্ছা হয় গলায় দড়ি দিয়ে মরি!	আমার আর ভালো লাগে না মশায়, এক সময় গলায় দড়ি দিয়ে মরতে ইচ্ছে করে!	আর ভালো লাগে না স্যার, এক সময়ে গলায় দড়ি দিয়ে মরতে চাই!

Chapter 7

Limitations, future research and synthesis of Our Research

Limitations of Our Research

Here are some constraints that we encountered when developing an authorship detection and imitation of writing personality in Bangla. First of all, an accurate authorship detection system is dependent on a wide-ranging and diverse dataset existing in Bangla. However, due to a lack of publicly accessible and structured datasets of Bangla text, it was complicated to compile a sufficiently large and diverse collection of Bangla text from numerous authors. Likewise, identifying distinctive features that capture an author's typing pattern in Bangla was challenging. While there are well-known methods for feature extraction in English, the same tactics can not be directly applicable to Bangla. Also, analyzing and processing vast amounts of Bangla text data can be computationally taxing, especially when working with intricate machine-learning techniques and vast feature sets. One notable issue is the need for a large-scale dataset to defend against adversarial attacks, which wasn't available in abundance in our research. Secondly, in the realm of poetry, it lacks significantly, as our focus is predominantly centered on prose, leading to a notable lack of attention and exploration within poems. Third, our model has further limitations as it focuses on only the Bengali language and is not designed to handle multiple languages, making it less versatile. Even though Bangladesh is well-versed in multilingual culture and language, we could have incorporated traditional Bangla in our methodologies but failed to implement approaches for regional, local, and ethnic languages. Further to this, we've essentially built our model based on renowned authors, but it falls short when it comes to lesser-known or local writers in Bangla literature. Besides that, we currently operate our system at the sentence level, and we haven't explored paragraph- or composition-level tasks, which might impact feasibility and adaptability for broader applications in literature. Again, during the style transfer phase, we've worked with Rabindranath's novels, but haven't ventured into other genres of his written work. Furthermore, during the building of datasets from scratch, we faced difficulties in obtaining accurate results. For instance, websites offering PDF-to-text conversion through online OCR (Optical Character Recognition) tools had limitations in terms of accuracy. Consequently, we had to do manual correction, which is again a less effective process, consuming time and energy. In addition to this, dataset creation posed challenges not only in the current era but also

in the old/historical era due to limited reliable online sources. For example, Begum Rokeya's writing from the old era is a notable one but has a shortage. Again, our evaluation reveals some sort of gap, such as the fact that we've predominantly used BLEU scores and loss curves in style transfer evaluation, lacking exploration with other metrics. Despite numerous attempts at parameter adjustments, our model often faces overfitting during training, presenting a persistent challenge. Lastly, we also faced hardware limitations, such as not having a high-spec GPU that posed challenges and resulted in substantial time consumption.

Future research and synthesis

There are some noteworthy future research benefits of this paper. First of all, there could be multi-dimensional opportunities for the improvement of privacy protection techniques and streamlining content verification processes. For instance, our authorship mimicry tool can be aligned with laws to tackle plagiarism and preserve original content. Further research could involve collaboration with legal experts to establish guidelines and policies to rectify fake content in the literary space. Moreover, bookstores and online platforms or publishers can use these models as a distinctive feature and might offer readers a tool to verify the authenticity of creative content. Secondly, the researcher can also explore the application of this model for cross-lingual authorship identification and imitation across different languages and cultural contexts. Thirdly, developing user-friendly interfaces may contribute to a more engaging online experience. This may allow interaction with chatbots or websites that replicate the writing style of their chosen authors. Furthermore, our model can extend its capabilities by collaborating with writing communities to generate longer and more coherent content, such as articles, essays, chapters, or even an entire book. This can broaden the utility of the model in various writing scenarios. Lastly, future researchers will be inspired by this thesis paper and continue their extended study on multilingual style transfer techniques, especially those of Bangla literature, plagiarism detection, content verification, author attribution, etc., based on the model we are using. As our study provides a handsome probability of future research opportunities, thus, it has intrigued us to continue.

Chapter 8

CONCLUSION

To put it concisely, the world of Computer Science has seen incredible progress lately, with exciting developments showcasing a wide range of capabilities. While current methodologies show proficiency in English, they lag when it comes to Bangla, particularly with complex multilingual datasets. However, the wave of progress has not left Bangla untouched, as NLP enthusiasts are dedicating themselves to closing deficiencies. Enthusiastic computer researchers are actively working to address any gaps or limitations present in the Bangla language domain. Their dedication is evident as they strive to establish new benchmarks and parameters, especially in the NLP field. Inspired by the dedication of others, our team has been motivated to bring forth unique ideas and contribute to this field of Natural Language and Processing. This commitment has led us to actively participate in shaping the trajectory of development for Bengali. Due to plagiarism, many legal issues have risen over the years which have become unsolvable in the context of Bangla since we were unable to do a proper authorship attribution. However, our thesis tends to serve as a solution for such predicaments by accessing a wide range of fine-tuned datasets that help to identify, learn, and generate full sentences from an original input text by preserving its semantic elements through various processes. Simply put, our distinctive model stands out by its ability to identify and understand users' writing patterns, extract them, and subsequently generate new content that replicates their individual styles. We have incorporated techniques to handle adversarial attacks, ensuring that our system remains resilient against any attempts to deceive or manipulate it. This includes distinguishing between false and authentic content, preventing any alteration that might compromise the integrity of the information. But to ensure the quality of our model, we shall also conduct comparison experiments to evaluate the performance metrics of our system. While conducting experiments, we have worked with the 4 types of classification - Rabindranath vs Non Rabindranath, Rabindranath era vs Current Era, Rabindranath vs Multi Author & finally Adversarial data. For all these categories of classification we obtained a classification report, a confusion matrix, a model accuracy vs epoch graph, a model loss vs epoch graph and finally an ROC graph by running 2 models simultaneously. One model was the Bidirectional LSTM and the other was BanglaBERT+LSTM. From the analysis of our results section, we can observe that the overall performance for both the models were quite good with an approximate average accuracy, recall, f1-score and precision of over 0.90 for all classifications. In most of the cases, there were also no signs of class imbalance which leads us to a more accurate comparison. The model was also able

to identify patterns in the data successfully during model training, but on the other hand, both the models start to overfit to the training data, thus, showing inclination in some of the training loss graphs. This happens when the model is able to fit the data perfectly but it is not able to generalize to new data anymore. Furthermore, we can also see that while text generation, by using the mT5 model and BangtaT5 model, we achieved quite a high average (around 50% and above) BLEU score for both models. This suggests that our model is able to translate and generate texts with high accuracy, less grammatical errors and even in some cases, almost better than humans. So to conclude, after taking into account all of the factors, it is safe to say that this thesis paper is expected to serve as inspiration for future researchers, motivating them to undertake further extensive studies on techniques for multilingual style transfer.

Bibliography

[1] Z. Zhang, Z. Han, L. Kong, *et al.*, "Style change detection based on writing style similarity," *Training*, vol. 11, pp. 17–051, 1970.

[2] J. Kabbara and J. C. K. Cheung, "Stylistic transfer in natural language generation systems using recurrent neural networks," in *Proceedings of the Workshop on Uphill Battles in Language Processing: Scaling Early Achievements to Robust Methods*, 2016, pp. 43–47.

[3] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu, "Text classification improved by integrating bidirectional lstm with two-dimensional max pooling," *arXiv preprint arXiv:1611.06639*, 2016.

[4] S. W. Jang, J. Min, and M. Kwon, "Writing style conversion using neural machine translation," *preprint*, 2017.

[5] 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)*, IEEE, 2018, pp. 1–6.

[6] Z. Fu, X. Tan, N. Peng, D. Zhao, and R. Yan, "Style transfer in text: Exploration and evaluation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[7] S. Abujar, A. K. M. Masum, S. M. H. Chowdhury, M. Hasan, and S. A. Hossain, "Bengali text generation using bi-directional rnn," in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, IEEE, 2019, pp. 1–5.

[8] S. Abujar, A. K. M. Masum, M. Sanzidul Islam, F. Faisal, and S. A. Hossain, "A bengali text generation approach in context of abstractive text summarization using rnn," *Innovations in Computer Science and Engineering: Proceedings of 7th ICICSE*, pp. 509–518, 2020.

[9] T. Gröndahl and N. Asokan, "Effective writing style transfer via combinatorial paraphrasing.," *Proc. Priv. Enhancing Technol.*, vol. 2020, no. 4, pp. 175–195, 2020.

[10] R. Kawser, *Bangla ranked at 7th among 100 most spoken languages worldwide*, Dhaka Tribune, Feb. 2020. [Online]. Available: https://www.dhakatribune.com/world/201648/bangla-ranked-at-7th-among-100-most-spoken.

[11] A. Khatun, A. Rahman, M. S. Islam, H. A. Chowdhury, and A. Tasnim, "Authorship attribution in bangla literature (aabl) via transfer learning using ulmfit," *Transactions on Asian and Low-Resource Language Information Processing*, 2020.

[12] K. Krishna, J. Wieting, and M. Iyyer, "Reformulating unsupervised style transfer as paraphrase generation," *arXiv preprint arXiv:2010.05700*, 2020.

[13] B. Liebl and M. Burghardt, ""shakespeare in the vectorian age"–an evaluation of different word embeddings and nlp parameters for the detection of shakespeare quotes," in *Proceedings of the The 4th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, 2020, pp. 58–68.

[14] K. Lin, M.-Y. Liu, M.-T. Sun, and J. Kautz, "Learning to generate multiple style transfer outputs for an input sentence," *arXiv preprint arXiv:2002.06525*, 2020.

[15] L. Xue, N. Constant, A. Roberts, *et al.*, "Mt5: A massively multilingual pre-trained text-to-text transformer," *arXiv preprint arXiv:2010.11934*, 2020.

[16] S. Zhou, "Research on the application of deep learning in text generation," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1693, 2020, p. 012 060.

[17] A. Bhattacharjee, T. Hasan, W. U. Ahmad, *et al.*, "Banglabert: Language model pretraining and benchmarks for low-resource language understanding evaluation in bangla," *arXiv preprint arXiv:2101.00204*, 2021.

[18] K. Carlson, A. Riddell, and D. Rockmore, "Unsupervised text style transfer with content embeddings," in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, 2021, pp. 226–233.

[19] M. R. Kibria and M. A. Yousuf, "Context-driven bengali text generation using conditional language model," *Statistics, Optimization & Information Computing*, vol. 9, no. 2, pp. 334–350, 2021.

[20] R. Singh, J. Weerasinghe, and R. Greenstadt, "Writing style change detection on multi-author documents.," in *CLEF (Working Notes)*, 2021, pp. 2137–2145.

[21] M. Toshevska and S. Gievska, "A review of text style transfer using deep learning," *IEEE Transactions on Artificial Intelligence*, 2021.

[22] E. Zangerle, M. Mayerl, M. Tschuggnall, M. Potthast, and B. Stein, *Pan20 authorship analysis: Style change detection*, 2021.

[23] M. Kowsher, A. A. Sami, N. J. Prottasha, M. S. Arefin, P. K. Dhar, and T. Koshiba, "Bangla-bert: Transformer-based efficient model for transfer learning and language understanding," *IEEE Access*, vol. 10, pp. 91 855–91 870, 2022.

[24] G. Ríos-Toledo, J. P. F. Posadas-Durán, G. Sidorov, and N. A. Castro-Sánchez, "Detection of changes in literary writing style using n-grams as style markers and supervised machine learning," *Plos one*, vol. 17, no. 7, e0267590, 2022.

[25] E. K. Tokpo and T. Calders, "Text style transfer for bias mitigation using masked language modeling," *arXiv preprint arXiv:2201.08643*, 2022.