Shahjalal University of Science and Technology

Department of Computer Science and Engineering

CSE 452

# A Modern Approach to Authorship Attribution in Bangla Literature

| | |
|---|---|
| Anisur Rahman | Aisha Khatun |
| Reg. No.: 2015331028 | Reg. No.: 2015331042 |
| $4^{th}$ year, $2^{nd}$ Semester | $4^{th}$ year, $2^{nd}$ Semester |

| **Supervisor** | **Co-Supervisor** |
|---|---|
| Ayesha Tasnim | Md. Saiful Islam |
| Assistant Professor | Assistant Professor |

6th February, 2020

# A Modern Approach To Authorship Attribution in Bengali Literature

A Thesis submitted to the Department of Computer Science and Engineering, Shahjalal University of Science and Technology, in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering.

By

Anisur Rahman
Reg. No.: 2015331028
$4^{th}$ year, $2^{nd}$ Semester

Aisha Khatun
Reg. No.: 2015331042
$4^{th}$ year, $2^{nd}$ Semester

Department of Computer Science and Engineering

**Supervisor**
Ayesha Tasnim
Assistant Professor

**Co-Supervisor**
Md. Saiful Islam
Assistant Professor

Department of Computer Science and Engineering

6th February, 2020

# Recommendation Letter from Thesis Supervisor

The thesis entitled *"A Modern Approach To Authorship Attribution in Bangla Literature"* submitted by the students

1. Anisur Rahman
2. Aisha Khatun

is under my supervision. I, hereby, agree that the thesis can be submitted for examination.

Signature of the Supervisor:
Name of the Supervisor: Ayesha Tasnim
Date: 6th February, 2020

Signature of the Co-Supervisor:
Name of the Co-Supervisor: Md. Saiful Islam
Date: 6th February, 2020

# Certificate of Acceptance of the Thesis

The thesis entitled *"A Modern Approach to Authorship Attribution in Bangla Literature"* submitted by the students

1. Anisur Rahman
2. Aisha Khatun

on 6th February, 2020, is hereby accepted as the partial fulfillment of the requirements for the award of their Bachelor Degrees.

Head of the Dept.
Mohammad Abdullah Al Mumin
Professor
Department of Computer Science and Engineering

Chairman, Exam Committee
Dr Mohammad Reza Selim
Professor
Department of Computer Science and Engineering

Supervisor
Ayesha Tasnim
Assistant Professor
Department of Computer Science and Engineering

Co-Supervisor
Md. Saiful Islam
Assistant Professor
Department of Computer Science and Engineering

# Abstract

Authorship Attribution deals with the task of creating an appropriate characterization of texts that captures the writing style of authors for accurate classification. With increased anonymity on the internet, this task has become increasingly crucial in various fields of security and plagiarism detection. Despite a few strands of work recently, Bengali lacks comprehensive research in this field mainly due to the scarcity of resources. In order to extract stylometric features from text, the smallest unit is character. In this report we start the exploration of authorship attribution with character level models and contrast them against traditional word based models. We show that the results with character based models are competitive with word based models but in terms of memory and time consumption for training, the character level models far outperform the previous models. The effect of pretraining character embedding was also analysed for the diverse Bengali character set in authorship attribution. It was seen that the performance had improved by up to 10% on pretraining showing that the character embeddings bore meaning information about the authorial writing style. Later in the report we employ a transfer learning approach to perform classification for authorship attribution in Bengali language. Language models are generally employed to estimate the probability distribution of various linguistic units, making them one of the fundamental parts of natural language processing. Applications of language models include a wide spectrum of tasks such as text summarization, translation and classification. We propose an approach for authorship attribution in Bengali based on AWD-LSTM architecture and transfer learning from language models in three steps. Comparison of three variations of the proposed models are shown, with the major difference being in tokenization - word, sub-word and character level tokenization. We also compare some previous state-of-the-art models and show that the transfer learning approach using sub-word tokenization pre-trained on News dataset performs better than other approaches. Specifically an increase of 18% and 6% on accuracy for the final target datasets is observed.

**Keywords:** authorship attribution, Bengali, deep learning, transfer learning, language models, convolutional neural networks, recurrent neural networks, LSTMs, character-level, natural language processing, universal language model, fine-tuning.

# Acknowledgements

We would like to thank the Department of Computer Science and Engineering, Shahjalal University of Science and Technology, Sylhet 3114, Bangladesh, for supporting this research. We are very thankful to our honorable supervisor Ayesha Tasnim and co-supervisor Md. Saiful Islam, whose contributions have been invaluable for our research.

# Contents

Abstract . . . . . . . . . . I

Acknowledgement . . . . . . . . . . III

Table of Contents . . . . . . . . . . IV

List of Figures . . . . . . . . . . VII

List of Tables . . . . . . . . . . VII

**1 Introduction** **1**

1.1 Motivation . . . . . . . . . . 3

1.2 Report Structure . . . . . . . . . . 5

**2 Related Works** **6**

2.1 On Author Attribution . . . . . . . . . . 6

2.2 On Embedding . . . . . . . . . . 7

2.2.1 Word Embedding . . . . . . . . . . 8

2.2.2 Character Embedding . . . . . . . . . . 8

2.3 On Transfer Learning . . . . . . . . . . 8

2.4 On Training Neural Networks . . . . . . . . . . 9

**3 Data Analysis** **12**

3.1 News Corpus . . . . . . . . . . 12

3.2 Wikipedia Corpus . . . . . . . . . . 12

3.3 Author Attribution Dataset(Dataset1) . . . . . . . . . . 13

3.4 Author Attribution Dataset(Dataset2) . . . . . . . . . . 15

**4 Phase 1: Character-Level Classification with CNN 16**

4.1 Overview . . . . . 16

4.2 Proposed Architecture . . . . . 17

4.3 Character Embedding . . . . . 18

4.4 Training the model . . . . . 18

4.4.1 Pre-training Embedding . . . . . 19

4.4.2 Performing Classification . . . . . 19

4.5 Experiments . . . . . 19

4.5.1 Word Embedding Model . . . . . 20

4.6 Results and Discussion . . . . . 20

**5 Phase 2: Universal Language Model Fine Tuning for Authorship Attribution 24**

5.1 Overview . . . . . 24

5.2 Proposed Architecture . . . . . 25

5.2.1 Language Model . . . . . 25

5.2.2 Classifier . . . . . 26

5.3 Tokenization . . . . . 26

5.3.1 Word level . . . . . 27

5.3.2 Sub-word level . . . . . 27

5.3.3 Character level . . . . . 27

5.4 Training . . . . . 28

5.4.1 Universal Language model . . . . . 28

5.4.2 Language model Fine-tuning . . . . . 30

5.4.3 Classification . . . . . 31

5.5 Experiments . . . . . 31

5.5.1 CNN-LSTM Word level classifier . . . . . 31

5.5.2 CNN Character level classifier . . . . . 32

5.6 Results and Discussions . . . . . 32

5.6.1 Model Effect . . . . . 33

5.6.2 Tokenization Effect . . . . . 34

5.6.3 Effect of number of Authors . . . . . 34

5.6.4 Effect of pre-training datasets . . . . . 36

5.7 Pre-trained Language Models . . . . . 36

**6 Phase 3: Transformer Based Transfer Learning for Authorship Attribution 39**

6.1 Overview . . . . . 39

6.2 Architecture . . . . . 40

6.2.1 Input/Output Representation . . . . . 40

6.2.2 Transformer Base and pretraining . . . . . 41

6.2.3 Classification . . . . . 41

6.3 Results and Discussion . . . . . 42

6.4 Further Works . . . . . 42

**7 Conclusion And Future Works 44**

7.1 Future Plan . . . . . 45

**8 Publications 46**

# List of Figures

2.1 Cyclical learning rate for various stages of training. . . . . . . . . . . . . . . . . . 9
2.2 Application of Discriminative Learning Rates [1] . . . . . . . . . . . . . . . . . . 10
2.3 Resetting Learning Rate after each Epoch [1] . . . . . . . . . . . . . . . . . . . . 11

3.1 Dataset1 Statistics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

4.1 Comparison of Word and Character embedding based models . . . . . . . . . . . 22
4.2 Effects of pretraining character embeddings. . . . . . . . . . . . . . . . . . . . . . 22

5.1 DropConnect Network . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
5.2 DropConnect Network . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26
5.3 Universal Language Model Fine Tuning Steps. . . . . . . . . . . . . . . . . . . . . 28
5.4 Text generation from different language models. . . . . . . . . . . . . . . . . . . . 29
5.5 Effect of tokenization on classification . . . . . . . . . . . . . . . . . . . . . . . . 35
5.6 Effect of word and sub-word tokenization on classification . . . . . . . . . . . . . 35
5.7 Performance of different models with increasing number of authors . . . . . . . . 37
5.8 Effect of dataset used in pre-training on classification . . . . . . . . . . . . . . . . 38

# List of Tables

3.1 News dataset statistics . . . . . 13
3.2 Authorship attribution Dataset1 statistics . . . . . 14
3.3 Authorship attribution Dataset2 statistics . . . . . 15

4.1 Performance comparision of different models with pretrained embedding . . . . . 20
4.2 pretrained vs non-pretrained comparison . . . . . 21

5.1 Universal Language model hyper-parameters . . . . . 29
5.2 Fine-Tuning Language model hyper-parameters . . . . . 30
5.3 Results of Classification . . . . . 33
5.4 Accuracy for various number of Authors . . . . . 36
5.5 Perplexity of Universal Language Models . . . . . 37

6.1 Results of Classification . . . . . 42

# Chapter 1

# Introduction

Authorship attribution is a type of complex classification task and is generally concerned with detecting the author of a piece of writing by analyzing its stylometric and linguistic features within a set of probable authors. This is different from authorship verification, which identifies if the document belongs to a certain author and authorship characterization, which reveals characteristics of an author from an unknown text. Anonymity is very common in recent times, especially due to widespread use of internet the use and misuse of anonymity has become an important factor to consider. The plethora of anonymous digital footprints makes authorship attribution indispensable in various fields and its applications are growing constantly. Application of authorship attribution covers many sectors such as forensic linguistics, plagiarism detection, computer security, criminal law, cyber-crime, literature, trading etc.

Texts can have two properties: the topic and the style of writing. The task of detecting the author depends mainly on capturing the elusive character of an author's style from their texts. To perform such task important features are extracted and classified using machine learning approaches[2]. Most stylometric studies use some items of language such as lexical and syntactic components. Among them, character n-gram has been used widely and shown to provide insights into the writing style of the authors[3]. Some research extracted semantic information using pre-trained word embeddings[4]. With increasing computing power, various deep learning models have also been successfully applied to the task, achieving impressive results[5, 6].

However, despite the current advancements in authorship attribution, sufficient work has not been done in Bengali. Some traditional approaches have been implemented including feature extraction [7] and different types of word embedding[8]. Very recently deep learning models were applied using character and word level tokenization[4]. Significant research work contributing to the implications of transfer learning for Bengali literature is not done yet to our knowledge. An elaborated study on the application of transfer learning for author detection in Bengali using language modeling[9] is presented in this report. We performed unsupervised training of the language model followed by authorship attribution specific fine-tuning and classification. Effects of various tokenization on this model are analyzed in terms of performance. The results demonstrate a clear superiority of the transfer learning based approach against the other traditional models. Concretely, the sub-word level model outperforms the other models in two target datasets. The proposed language model based architecture outperforms former state-of-the-art models by a significant margin of 6% to 12%. **All models, along with the datasets created have been released for public use**.[1]

Dataset1(described in 3.3) is imbalanced, therefore resembles a real-world scenario more closely where the text samples for each authors may not be found in equal amounts. To employ transfer learning, language model is trained on two different datasets - Wikipedia and News corpus. All Datasets are described in detail in section 3. Besides, experiments are performed with various numbers of authors to show the model's ability to perform on lesser information yet larger number of authors to detect from.

In this report, our work is divided into three phases. The first phase consists of initial authorship attribution experimentation with character level embedding pretraining. Phase 2 contains the bulk of our work with transfer learning based approach. In this phase we use various types of tokenizations and pretraining datasets and compare all approaches. In phase 3 we start using transformer based architecture, specifically BERT for our specific task. A detailed outline of the report can be found in the report structure section.

[1]https://github.com/tanny411/Authorship-Attribution-using-Transfer-Learning

## 1.1 Motivation

The core motivations of our research on the topic of Authorship Attribution are as follows:

- □ **Enrich Bengali research for authorship attribution task:** Although Bengali language is very prominent with the number of speakers around worldwide, the amount of substantial research on the field of authorship attribution is scanty. We wanted to bridge this gap with a new approach. Our hope is that this research will be a pioneer epitome for transfer learning or deep learning approach on authorship attribution for Bengali literature. This research will have its due impact into the advancement and further extension for this particular area of research.

- □ **Attribution to a Forensic Level:** Previous works, to our inspection, was lacking a specific focus in overall big picture of the task. Whenever a research was performed, there was some level of inconsistency between the dataset and classification task. When overall accuracy was good, the dataset seemed to be inconsistent. When the dataset was standard, performance was not convincing enough. All in all, most of the models currently developed could not be convincingly implemented into practical applications. We wanted to investigate deep learning models with a view to take the authorship attribution in Bengali Literature to a forensic level. For which, the research should be very efficient in handling the diversity error of the authors.

- □ **Introducing Deep Learning in the task:** Works done so far mainly includes Support Vector Machines, Naive Bayes Classifiers and multiple other probabilistic and non-probabilistic models in Bengali Language for authorship attribution. This approaches is not very efficient for some scenarios and handling the complexity which may arise due to the subtle nuances of the problem domain. However, the touch of deep learning in this section range from very little to nearly non-existent.

- □ **A Benchmark in Diversity:** Classifiers like the Support Vector Machine performed with impressive accuracies of above 90% in classification of authors when the diversity of categories and topics was low. Whenever the number of authors was increased, the performance of traditional models dropped drastically. Our approaches, described in the chapters ahead, try to tackle this problem of diversity head-on.

## 1.2 Report Structure

The most important part of this report is the Phase 2 section. Almost all the substantial work is explained from the very beginning to the end of that chapter. Phase 1 contributes to a small part of this research as it introduces character-level embedding, a very new concept on Bengali text-classification task yet unexplored. It tries to break a text to its most core part, character, from which it tries to evaluate the sequence of those for finding the hidden patterns and meanings.

- The first section of the paper includes a strong background study on the fundamentals of authorship attribution.
- The datasets are mentioned in full details with necessary statistics in data analysis portion next. These include both pretraining text corpus and authorship attribution datasets.
- Phase 1, a supplementary part of this research, includes the concepts of character-level embedding to which we compared the findings and performance of word-embedding.
- Phase 2, the main part of this research, includes the major portion of the research, where we surpass our own highest performance in this task. Phase 2 is a complete portion in itself and contributes almost to the entirety of the research.
- Phase 3, a newer path for the task at hand using transformer based transfer learning. We show our initial findings and points out potentials paths to explore from here.
- We then describe about our actual research contribution and conclude with some future scopes for research which we will try to work on the coming days.

# Chapter 2

# Related Works

## 2.1 On Author Attribution

The task of authorship attribution is an important research topic and has been prevalent for quite a long time. Research in authorship attribution relies on detecting the styles of authors through stylometry analysis assuming that the authors are subconsciously using homological idiolect in their writings. To detect these patterns different kinds and sets of feature extraction were implemented. The features were mainly classified as lexical, character, syntactic and semantic. Besides, most of the past researchers dealt with small collections where each author may have been strongly inclined towards particular topics. This made the authorship attribution task easier and bordering on topic classification. Later work is focused on character n-grams as they are able to pick up author nuances including lexical and syntactic information [3]. Combination of lexical and syntactic features provides valuable information and it has shown improvement in performance [10]. Such combinations can be helpful to boost the performance in cross-topic authorship attribution and single-domain attribution [11].

Marching on with the advancement of deep learning, a large body of work is available on authorship attribution and stylometry using various deep learning models. For example, multi-headed recurrent neural network character language model were used that outperformed the other methods in PAN 2015 [12]. Some works used syntactic recurrent neural network which learns document representations from parts-of-speech tags and then attention mechanism to detect the authorial writing style [5].

Convolutional neural networks have also been employed for this task. Impressive performance was achieved by using character-level and multi-channel CNN for large scale authorship attribution [6]. Character n-grams help identify the authors of tweets and CNN architectures capture the character-level interactions which can represent patterns in higher levels, thus detecting distinct styles of authors [13]. Combination of pre-trained word vectors with one hot encoded POS tags are also used [14]. Others investigate syntactic information in authorship task by building separate language models for each author using part-of-speech tags besides word and character level information [15].

Despite the amount of work being done in English and other languages for authorship attribution, not much has been accomplished in Bengali language, especially with transfer learning for authorship attribution. Hand-drawn features such as POS tag count, word length, word frequency etc were used with a 4 author dataset [7]. High accuracy was achieved using multiple features and cosine similarity [16]. SVM was employed on a dataset of 6 authors [17]. Multilayer perceptrons were also used [18]. The effects of word embeddings were analyzed on authorship attribution [8]. They showed that the fastText skip-gram with a CNN model beats other models on a 6 author dataset. The distinct nature of Bengali text and its high vocabulary count serve to make training using traditional methods complicated. The application of transfer learning and its effects on authorship attribution are yet to be analyzed.

## 2.2 On Embedding

Embeddings are defined as mappings of tokens to a high dimensional space. These vectors contains information about the semantic, syntactic, and morphological aspects of the token. Tokens can be anything from character, word to pieces of word or entire sentence. The tokens are usually input to any machine learning system as these embeddings. The system thus finds patterns and performs various tasks on the given text.

### 2.2.1 Word Embedding

Word embeddings are often learned in an unsupervised manner such as Continuous Bag-of-Words(CBOW), Skip-Gram models and co-occurance method. These are implemented as word2vec, fastext and Glove. CBOW learns embeddings through prediction of the current word based on its context(surrounding words). Skip-gram learn by prediction of the context given a word. Glove on the other hand contructs co-occurance matrix of words and learn the embeddings. Embeddings are used widely and their use has achieved numerous breakthroughs in NLP. Word embeddings and convolutional models were used together to improve over other methods [19], in sentiment analysis [20] and pretrained embeddings are also used in different tasks [21, 22]. Various researches have broken down words into subword and character levels. [23] creates subword embedding from counts of character n-grams.

### 2.2.2 Character Embedding

Character level embeddings are used as is, or to create embeddings in higher orders such as for words, subwords etc. They have been used in POS tagging [24], dependency parsing [25], and language modelling [24]. They are also used in machine translation by word [26] or by character [27]. Pure character level classification was first demonstrated using CNN model [28] and it was shown that such models can significantly surpass state of the art performances [29].

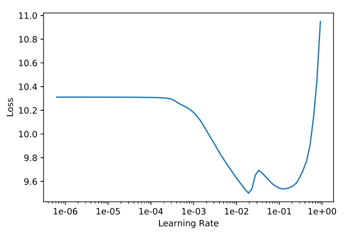
## 2.3 On Transfer Learning

Transfer learning is the process of reusing a model trained on an initial task as a starting point for a different task. Deep learning approaches have been using this method to boost model performance and save an enormous amount of time on various tasks of computer vision and natural language processing. Transfer learning has been prevalent in computer vision for a long time and pre-trained models on very large datasets are readily available. With this respect, natural language processing mostly uses word embeddings as a source of transfer learning which aims at only the first layer of the model. Some approaches combine multiple derived embeddings at different layers [30]. Nevertheless, the need for training from scratch remains. The idea of using transfer learning from language models has been approached [31] but not widely adopted due to its need for large-scale datasets. To address these issues a method called ULMFiT was proposed that successfully enables robust transfer learning in any NLP task [9]. Since then, many NLP tasks have been able to outperform SOTA on respective datasets using successful transfer learning in this approach.

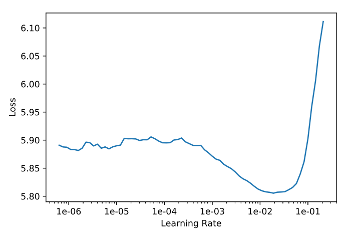
## 2.4 On Training Neural Networks

In recent research, training neural networks efficiently has become a cumulative task of proper architecture selection, suitable techniques application and fine-tuning. This section briefs the techniques used in this paper.

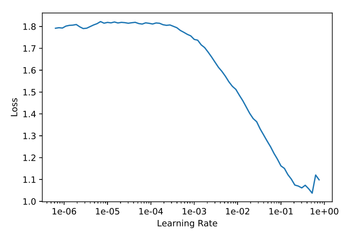
**1cycle policy:** Learning rate is one of the most important hyper-parameter that drastically effects experiments and has been manually tuned for quite some time. An impressive approach was published in this regard [32]. In this method, a single trial is run over the dataset starting with a low learning rate and is increased exponentially batch wise. The loss for each value is recorded. The point of decreasing loss and yet high learning rate are selected as the optimal values. Examples of learning rate can be observed from Figure 2.1.



(a) Universal Language Model



(b) Fine-tuning language model



(c) Classification

Figure 2.1: Cyclical learning rate for various stages of training.

**Discriminative learning rate:** Most-often, the chosen learning rate is fixed throughout the training process and among all the layers. The various layers in a deep learning model contain different levels of information and sometimes it may be necessary to allow for some information to be preserved or changed slowly compared to the others. A technique known as discriminative learning rate [9] solves the problem. This method ensures that the later layers of a neural network train faster than the base layers. Building deep learning models on top of the embedding layers has shown promising results in the recent past, which implies the later layers of the model need to be somewhat modified. Using discriminative learning rates, therefore, serves this purpose. Figure 2.2 provides an illustration of the discriminative learning rates.

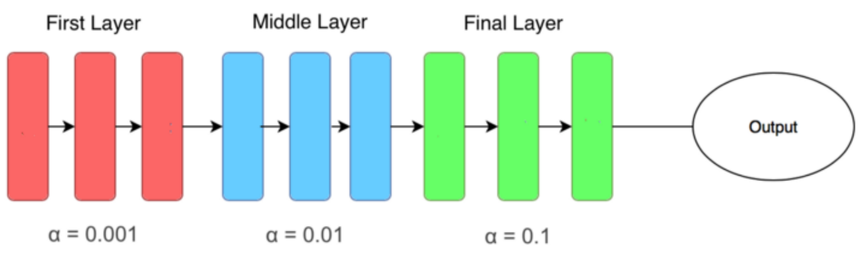


Figure 2.2: Application of Discriminative Learning Rates [1]

**Gradual unfreezing:** In the case of transfer learning, when a task specific model is trained on top of a pre-trained model, weights of all the layers are altered together. This method carries noise back to the base layers from the newly attached(randomly initialized) layers. To prevent such abrupt alteration and therefore catastrophic forgetting, we employ gradual unfreezing [9]. This method first freezes the initial layers and trains only the last layers, then the pre-trained layers are unfrozen one by one and trained further to tune to the problem-specific domain. Thus, the learned information of the pre-trained model is used efficiently.

**Slanted triangular learning rates:** The original concept of stochastic gradient descent [33] tries to employ the idea that the neural networks should be getting closer to the global minimum value of the loss. As the global minimum for loss gets closer, the learning rate should decrease quite obviously. If it doesn't, the system may fall in an infinite loop as it jumps from one side of the global minima to another because it keeps subtracting the multiple of the large learning rate selected. Slanted triangular learning rate (STLR) linearly increases the learning rate to quickly converge and then linearly decays it according to an update schedule in-order to tune the parameters [9].

**Stochastic Gradient Descent with restarts:** It is highly likely the training phase of the neural network will encounter local minima in the loss curve rather than the global minimum. In such cases, the model may get stuck at the local minima instead of reaching for the global minima. A technique that approaches this problem involves sudden increase in learning rates in hopes that it will 'jump' over the local minima if there are any. The process is called stochastic gradient descent with restarts and was introduced in [33]. SGDR suddenly increases the learning rate, and then decreases it again by cosine annealing. Figure 2.3 describes how learning rates are restarted after every epoch to avoid the problem.

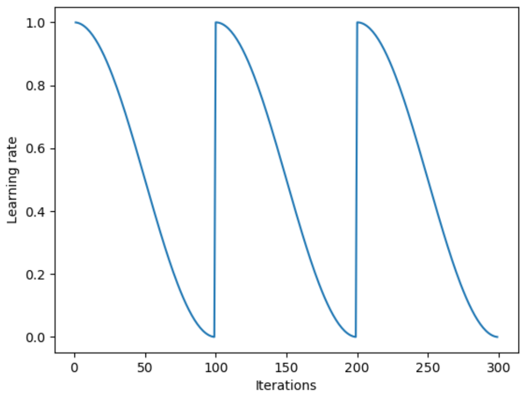


Figure 2.3: Resetting Learning Rate after each Epoch [1]

# Chapter 3

# Data Analysis

The scarcity of standard benchmark datasets in Bengali language makes the task of authorship attribution quite difficult. To overcome this issue we build a dataset and use it alongside a previously published dataset in this field. In total 4 datasets were used in different steps of the experiments.

## 3.1 News Corpus

This corpus is made of various Bengali newspaper articles. It includes articles on 12 categories. This corpus is much bigger in size compared to the previous works in Bengali. There are 28.5+ million word tokens in this whole corpus and the number of unique words is 836509 forming around 3% of the total vocabulary. Table 3.1 shows the details of the dataset along with some basic statistical measures.

## 3.2 Wikipedia Corpus

We collected a subset of Bengali Wikipedia text for the purpose of our experiments. To create the Wikipedia dataset, we have collected the Bengali wiki-dump[1]. The files are then merged and each article is selected as a sample text. All HTML tags are removed and the title of the page is stripped from the beginning of the text. This dataset contains 70377 samples with a total number of words being 18+ million. The entire dataset has 1289249 unique words, which is 7% of the total

[1]collected on 10th June, 2019

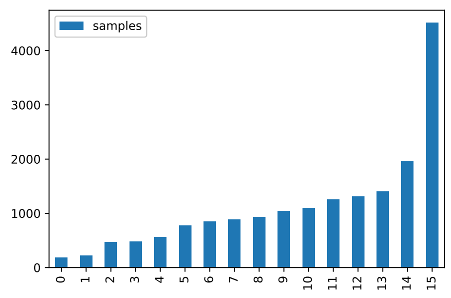
Table 3.1: News dataset statistics

| Category | Samples | Word count | Unique word |
|---|---|---|---|
| opinion | 8098 | 4185472 | 243968 |
| international | 5155 | 1089780 | 86852 |
| economics | 3449 | 909648 | 58932 |
| art | 2665 | 1312571 | 154869 |
| science | 2906 | 697899 | 76755 |
| politics | 20050 | 6167418 | 196541 |
| crime | 8655 | 2016342 | 128308 |
| education | 12212 | 3963695 | 225348 |
| sports | 11903 | 3087029 | 174677 |
| accident | 6328 | 1086791 | 77171 |
| environment | 4313 | 1347509 | 103783 |
| entertainment | 10121 | 2669492 | 204902 |
| **Average** | 7988 | 2377803 | 144342 |
| **Total** | 95855 | 28533646 | 836509 |

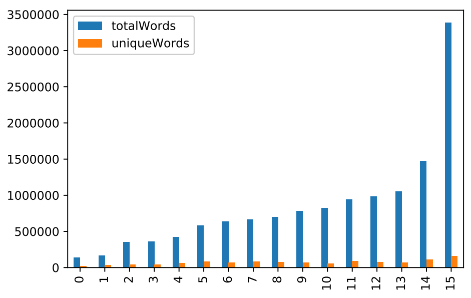
vocabulary. This makes the Wikipedia dataset more varied in terms of the types of words being used, compared to the News dataset.

## 3.3 Author Attribution Dataset(Dataset1)

This dataset has been collected from an online Bengali library and contains literary works of different Bengali famous writers [4]. It contains novels, stories, series and other works of 16 authors. Each sample document is created with 750 words. In Table 3.2 details are shown about the dataset. This dataset is imbalanced as apparent from Figure 3.1. We create multiple balanced subsets of this dataset for various number of authors in-order to measure model stability with increasing author numbers. The datasets are truncated to the minimum number of samples available per author.



(a) Sample distribution



(b) Words distribution

Figure 3.1: Dataset1 Statistics

Table 3.2: Authorship attribution Dataset1 statistics

| **Author** | **Samples** | **Word count** | **Unique word** |
|---|---|---|---|
| candidate 01 | 185 | 138750 | 20568 |
| candidate 02 | 223 | 167250 | 33124 |
| candidate 03 | 469 | 351750 | 44477 |
| candidate 04 | 476 | 357000 | 43864 |
| candidate 05 | 562 | 421500 | 62485 |
| candidate 06 | 775 | 581250 | 84311 |
| candidate 07 | 849 | 636750 | 67579 |
| candidate 08 | 888 | 666000 | 84888 |
| candidate 09 | 931 | 698250 | 76071 |
| candidate 10 | 1048 | 786000 | 69182 |
| candidate 11 | 1100 | 825000 | 53163 |
| candidate 12 | 1259 | 944250 | 89956 |
| candidate 13 | 1312 | 984000 | 78717 |
| candidate 14 | 1408 | 1056000 | 69648 |
| candidate 15 | 1963 | 1472250 | 109230 |
| candidate 16 | 4518 | 3388500 | 161893 |
| **Total** | 17966 | 13474500 | 590660 |
| **Average** | 1122.875 | 842156.25 | 71822.25 |

## 3.4 Author Attribution Dataset(Dataset2)

It is a 6 Author dataset collected and analysed by Hemayath et al [8]. The total number of words and unique words of this dataset are 2304338 and 230075 respectively. The data are obtained from different online posts and blogs. This dataset is balanced among the 6 Authors with 350 sample texts per author. Total word count and number of unique words per author are shown in Table 3.3.

Table 3.3: Authorship attribution Dataset2 statistics

| **Author** | **Samples** | **Word count** | **Unique word** |
|---|---|---|---|
| fe | 350 | 357453 | 53613 |
| ij | 350 | 391033 | 72034 |
| mk | 350 | 377100 | 47669 |
| rn | 350 | 231396 | 50029 |
| hm | 350 | 555710 | 72624 |
| rg | 350 | 391646 | 58071 |
| **Total** | 2100 | 2304338 | 230075 |
| **Average** | 350 | 384056.33 | 59006.67 |

# Chapter 4

# Phase 1: Character-Level Classification with CNN

## 4.1 Overview

Traditionally texts are represented using independent features such as lexical n-gram or frequency based representation. Generally, these approaches are incompetent for taking the similarity or context of the words into account as these features are mostly independent and do not compensate for the diversity. So, the semantic values of the words might be lost, which is problematic. One plausible solution to this problem is to use word embedding, also generally known as distributed term representations. It encodes semantic similarity from their co-occurrences. Chowdhury[8] experimented with the effectiveness of word embedding in authorship attribution for Bangla language experimenting with various architectures.

So, in the rudimentary phase of our research we decided to experiment with character embeddings as this type of work was not introduced for Bangla. Character CNN was first introduced by zhang [28] for text classification task. From the empirical experiment of Sebastian[6] and Jozefowicz[29] character level NLP seems to be very promising for many practical uses. Although, one may think that characters itself of a language does not bear any meaning, but some research papers [34] illustrates that character level models can capture the semantic properties of the overall text. Character level models also gives better performance for out-of-vocabulary words, misspelling etc and provide an open vocabulary. When working with word embeddings, usually the dimension are much higher compared to character embedding. So, this gives a huge boost in removing the bottlenecks in training tasks and also gives a huge advantages on computational as well as space complexity.

This chapter describes full experimental setup and the findings of our experiment with character embeddings implemented for authorship attribution in Bengali language. A comparison with word embeddings in performance is also stated. To better the performance of the proposed model, the idea of pre-training was also explored. Outcomes of all the experiments are summarized in result and discussion section.

## 4.2 Proposed Architecture

Character-level CNN can be used instead of words for classifications[28]. CNN can work even without the syntactic and semantic structure of the language. Which makes these models language independent because character set are fixed for a language. In this experiment we use CNN with different architectures to get the best performance successfully extracting character level features of the text. We worked with 3 different dataset to preserve the diversity effect. This architecture were later used for preparing the pre-trained character embeddings for classification. To describe the architecture in simple terms, it is a deep neural network starting with 4 convolutional layers, each followed with a maxpool layer of kernel size 3. In the convolutional layers the number of filters increase while decreasing the kernel size at each layer as standardized in computer vision. The kernel sizes are 7,3,1 and 1 respectively. The number of filters are 64,128,256 and 256. First part is an embedding layer where each character is represented as a vector of length $||V||$ equal to the alphabet size. The convolutional layers are stacked with a fully connected layer of 512 activation nodes, activation function ReLU and dropout. Finally the output layer uses softmax to calculate the classification probabilities. Adam optimizer is used along with categorical cross-entropy as the loss function.

## 4.3 Character Embedding

In character embedding the main goal is to turn characters into meaningful numerical representation as opposed to words usually in form of a vector. these vector can represent the correlation between different characters and it can be further extended to the correlation among group of characters even in much bigger context such as words, sentences, documents etc. Such abstraction can be a solution to fit misspelled words, rare or new words, slang or emoticons. Variation within different words and parts of speech can also be handled with grouping of characters. Limitations that came with vocabulary handling is now taken care of. There is no bottleneck for out of vocabulary words as characters are the smallest unit for wording. They can make up any word. Whereas, in word embedding we had this problem. When a word appeared which is not presented in the vocabulary set it was simply ignored or given weak representation. Another significant improvement is the vocabulary size. Instead of having a very large vocabulary of words, character embeddings have a fixed number of characters which is significantly smaller and thus greatly reduces model complexity and the number of parameters. Furthermore, characters can be fit into a very small vector whereas for words it could go up to 300 or more. The simplest way to represent a character is to use one-hot encoding. We use one-hot encoding as a baseline for comparison of pre-trained embeddings with the size of the alphabet. It can be initialized randomly also and the size be treated as a hyper-parameter for further tuning.

## 4.4 Training the model

The alphabet size during this experiment was 253. The embedding vector size for characters was equal to alphabet size. The 253 different characters are consisted of english letters(capital and small) and digits, Bangla letters and digits, Bangla vowel symbols, and various other punctuation and symbols. Two sets of embeddings were created for the character set for comparative training. First one being hot encoding and the other is pre-trained embeddings. Training was done in two phases as stated below:

### 4.4.1 Pre-training Embedding

To make pre-trained character embeddings the architecture mentioned above was used on news dataset. This is in contrast to usual ways of learning embeddings. No separate model was used like[35] to learn the embeddings. Instead, already available classification task on a marginally large dataset learns character embeddings for its own purposes. Those embeddings can be used as initialization for the author attribution task, which has a smaller dataset compared to the former, thus giving it an initial boost. The model was trained with a learning rate of 0.001 and decay of 0.0001. Maximum length of each text sample was set as 1000 and batch size as 80. A dropout rate of 0.5 was used in the fully connected layer to prevent over-fitting. They are then extracted and used for the task of author attribution in our case.

### 4.4.2 Performing Classification

Final classification for authorship attribution was performed in two separate process. One with one-hot encoding and the other is with pre-trained embedding. A dropout probability of 0.7 was set in the fully connected layer and trained with batch size 128. Maximum length of each text was set to be 3000 characters. The experiment was done on two author attribution datasets separately. The first dataset had 6 authors[8] and the other one was our own curated dataset with maximum 14 authors.

## 4.5 Experiments

We showed the impact of pre-trained character level embedding in terms of accuracy juxtaposing to without pre-training model. Comparison between word embedding and character embedding is also an important aspect of out experiment. All models are compared for increasing number of authors(classes) on the corpus mentioned to asses the quality of the models. To equate the dataset number of samples per class is truncated to the minimum among the whole subset. We propose a model for word level classification mostly similar to our Char-CNN model. The model used for performance analysis is as follows:

### 4.5.1 Word Embedding Model

This model has close resemblance to the proposed CharCNN model except for a few differences to tune with the word level version of classification. The model has 2 convolutional layers with the kernel sizes 7,3 and number of filters are 128,256 respectively for each layer.Each layer followed by a maxpool layer. The model is initialized with pretrained word embeddings from word2vec and fastext, both CBOW and skipgram versions. The convolutional layers are stacked with an LSTM layer of 100 neurons and a fully connected layer of 512 activation nodes both with dropout to prevent overfitting. Finally softmax layer is used to provide the classification probabilities. It is trained for 10 epochs with a learning rate of 0.001 with Adam optimizer, batch size is 32 and 750 words per sample are used as input to models.All the word level models have a vocabulary size of 60000 and word embedding vector of size 300.

## 4.6 Results and Discussion

The summarized result of characterCNN with word embeddings are shown in the table 4.1 . These accuracy are obtained with pre-trained embeddings. Because the datasets were balanced, the comparison of accuracies is sufficient.

Table 4.1: Performance comparision of different models with pretrained embedding

| #Authors | 6[8] | 6 | 8 | 10 | 12 | 14 |
|---|---|---|---|---|---|---|
| #samples/author | 350 | 1100 | 931 | 849 | 562 | 469 |
| Char-CNN | 83 | 96 | 92 | 86 | 75 | 69 |
| W2V(CBOW) | 65.3 | 97 | 82.8 | 83.3 | 76.4 | 71.8 |
| fastText(CBOW) | 65 | 73 | 58 | 35.7 | 37.31 | 40.3 |
| W2V(Skip) | 79 | 94 | 91.1 | 85.4 | **82.2** | 78.6 |
| fastText(Skip) | **86** | **98** | **95.2** | **86.35** | 80.9 | **81.2** |

When pre-training character embedding was used models were initialized with already learned weight vectors instead of random values or only one-hot encoding. This gives the model a initial direction and boost for achieving better result. Table 4.2 shows the impact of pre-trained character embedding on the proposed model.

From the indications of table 4.1 we can see that Skip-gram implemented by fastText performs better for the given dataset. One reason behind that is fastText works at a sub-word level and

Table 4.2: pretrained vs non-pretrained comparison

| #of Authors | 6[8] | 6 | 8 | 10 | 12 | 14 |
|---|---|---|---|---|---|---|
| #of samples/class | 350 | 1100 | 931 | 849 | 562 | 469 |
| Pretrained Embedding | 83 | 96 | 92 | 86 | 75 | 69 |
| Not pretrained | 71 | 95 | 82 | 83 | 66 | 59.5 |

extracts better information and styles from the text than word level. However, Character level also performs comparatively with the sub-word level model(fastText) when the dataset is big enough. As the number of author increases so does the number of samples to per author because of truncation. It becomes difficult for the character level model to gather necessary information from this small dataset. Even then, it was the second best performing model in the list next to fastText. So, with bigger dataset, character level model is capable of performing significantly better as shown by zhang[28] in his research work. But there are other benefits to character level models. For example, in terms of number of parameter, character level model is definitely much superior to its word level counter parts. The embedding vectors for the word level models is of size $embeddingvector * vocabularysize$. i.e. 300 * 60000. On the other hand the character embedding matrix is of size 253*253 given that we initially used one-hot vectors. This size can also be reduced to as low as 253*16 as were done in some research[29]. Another aspect that should taken into consideration is the amount of time the model consumes in training. In case of word embedding models, to get the desired result an LSTM layer should be added to include time series information in the model and a pure CNN does not perform satisfactorily. Although it improves the accuracy for word embedding model but with the cost of taking more time to train. On the other hand, the character level model works significantly well with only using convolutional layers and the time complexity reduces up to even 8 times which is pretty amazing. As stated by zhang[28] in his research, ConvNets with character embedding can completely replace words and work even without any semantic meanings. Therefore, convolutional layers are able to extract whatever information necessary for the purposes of author attribution given enough data.

As for the experiment on usefulness of pre-trained character embedding, from the table 4.2 we can see that using a pre-trained embedding gives better accuracy across datasets and different number of authors regardless of the amount of training data available for each author. This

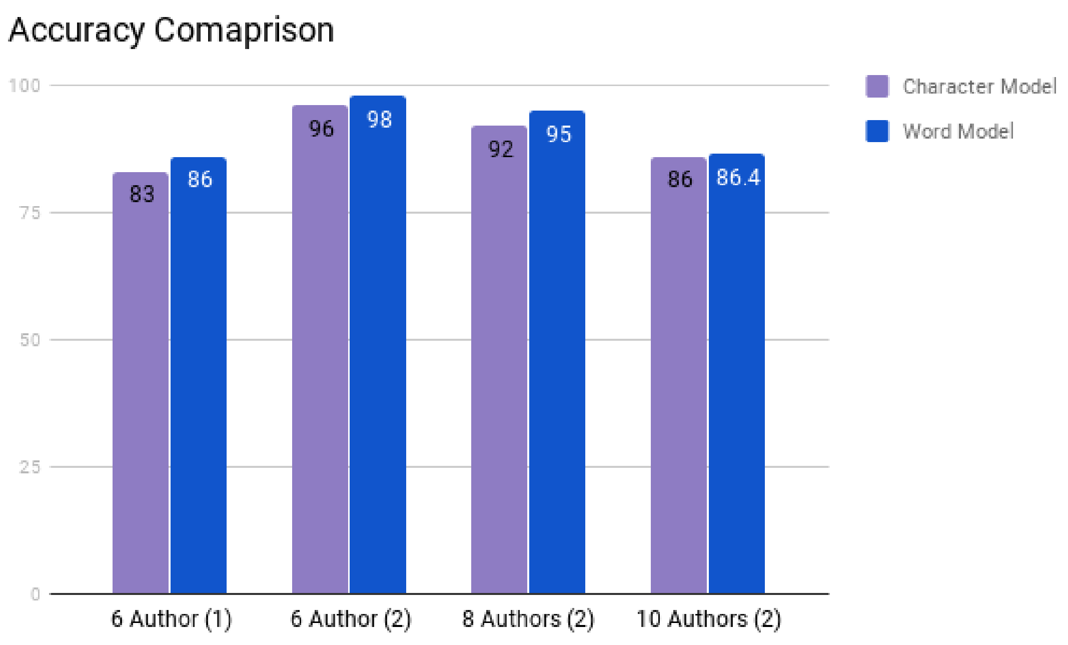


Figure 4.1: Comparison of Word and Character embedding based models

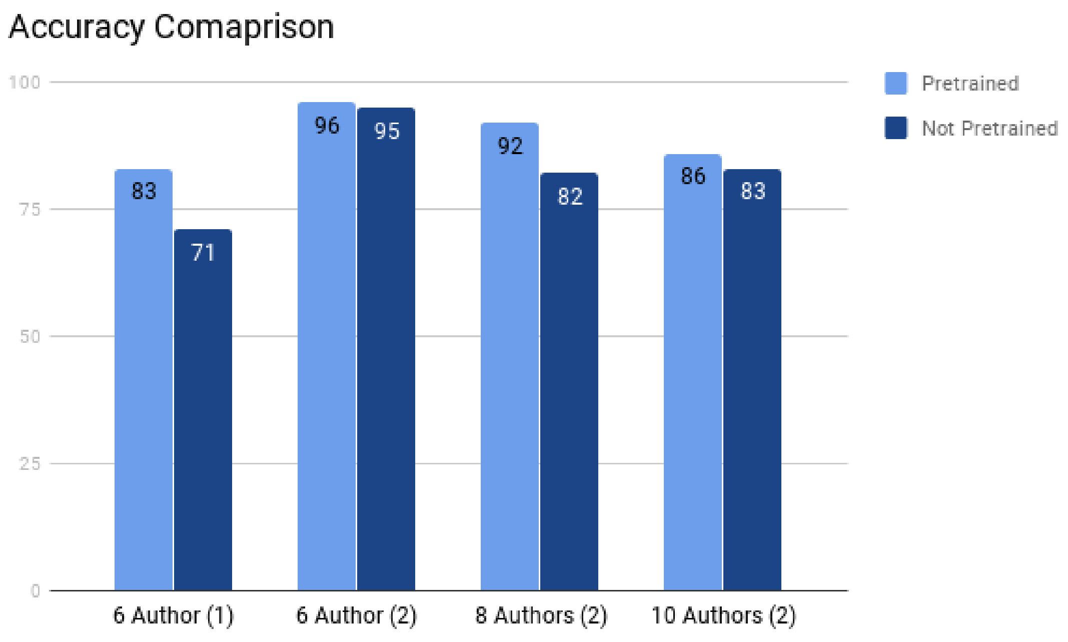


Figure 4.2: Effects of pretraining character embeddings.

shows that these naively learned embeddings contain valuable information that can be easily applied to various tasks of the language including author attribution. It also increases performance by significant margin. These numerical representations of character in the embeddings contain information about morphology, syntax of the language among other things. So, this can be inferred that such embeddings can be learned from any task and applied to other tasks as a form of transfer learning, given that the set of alphabet remains same.

# Chapter 5

# Phase 2: Universal Language Model Fine Tuning for Authorship Attribution

## 5.1 Overview

The traditional approaches of authorship attribution in Bengali Language, even in cases of deep learning methods just implement a classification model with word representations or other extracted features, taking very little regard of the language on its own. In this phase of our research, our approach was to make the model learn the language, and then perform the requires tasks, which includes pre-training a model and then performing classification in 3 steps. The idea of using such a setup is much like the traditional approach of using a single word-embedding layer, only here the language model is a multi-layered deep neural network on its own, holding much more information in its weights than a single layer of embedding matrix would [9]. The general idea of this research, however, is to capture the essence of Bengali language first, then proceed to the more granular task of classification with the help of language models. After the training phase, we test with some examples and find that the models created in our experiments can actually complete entire Bengali paragraphs with very little grammatical or semantic flaws. So, it is safe to assume that the weights of the language models hold enough relevant information to understand the patterns at which Bengali words appear one after another as opposed to single word-embedding layer which only represents relationship among words.

## 5.2 Proposed Architecture

### 5.2.1 Language Model

The patterns in the writings of authors recur and this nature of the task suggests the use of a recurrent neural network(RNN). In this case, we employed a special variant of the RNN, called the AWD-LSTM [36]. It stands for ASGD(Average Stochastic Gradient Descent) Weight-Dropped LSTM. This model provides special regularization techniques such as drop-connect and other optimizations that make it a suitable choice for generalizing context and language modeling. In traditional models, overfitting is a major issue. Drop-connect handles this issue by randomly selecting the activation subset on the hidden-to-hidden weight matrices. This preserves the RNNs ability to remember long-term dependency yet not overfit. Figure 5.1 provides a graphical example of the drop-connect network.

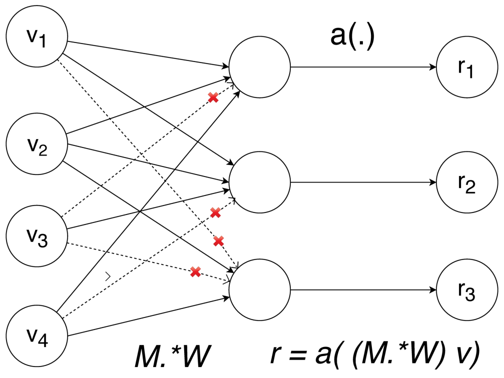


Figure 5.1: DropConnect Network

Among some other techniques is the generation of dropout mask on the first call namely the variational dropout [37]. Divisibility problem of elements is reduced by variable length backpropagation. Embedding Dropout [37] and reduction of embedding size all helped the architecture in achieving SOTA in language modelling [38] and therefore is the chosen for successful transfer learning [9].

The encoder network consists of an Embedding layer with an embedding size of 400, followed by 3 regular LSTM layers each with 1150 hidden nodes. It has a few short-cut connections and numerous drop-out hyper-parameters but does not use attention mechanism. The decoder is formed by a softmax linear layer, that provides the probabilistic estimations for the next word over the vocabulary. We used Adam optimizer [39] and flattened cross-entropy loss function. A simplified diagram of the architecture is shown in Figure 5.2 with an example text.

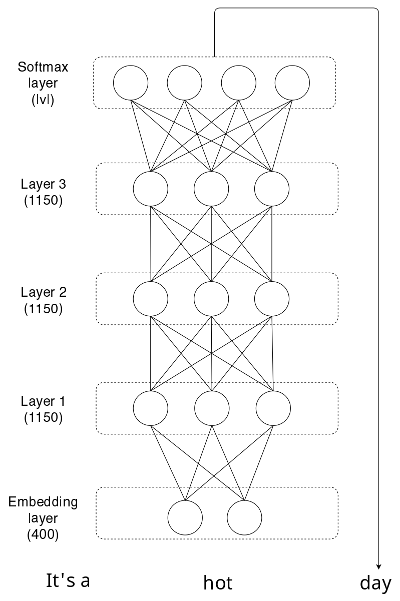


Figure 5.2: DropConnect Network

### 5.2.2 Classifier

The classifier is built on top of the language model by only changing the decoder part of the model. Two linear layers are added with batch normalization and dropout [9]. Activation function ReLU is used for intermediate layers and softmax for the output layer provided the necessary probability distribution to predict among the given authors.

## 5.3 Tokenization

Traditionally tokenization in Bengali has been carried out at word level, separating words by spaces. This includes separating punctuation or special characters as separate tokens. But Bengali language has some distinct characteristics which make mere separation by words less meaningful. Lemmatization and stemming provide a way for removing inflection, but this process remains arduous and only basic rule based systems are available to perform this task in Bengali. Besides, the removed parts of the words are not always completely meaningless and may provide some meaning in terms of gender, person, case, number or animacy. Information provided by declension is also a necessary part of the language to consider. With this end, besides word-level basic tokenization, we also perform sub-word and character level tokenization and compare the various methods' effect in the final result. All these methods are described elaborately in the following subsections.

### 5.3.1 Word level

Word level tokenization has been performed mainly by considering words separated by spaces as separate tokens. Besides, words that occur less than 3 times are discarded considering them as rare words, names or misspellings. We select the most frequent 60,000 words as the vocabulary to proceed to the network. Unknown words are replaced with the <unk> token. Some specialized tokens are also added such as the beginning of a string <bos>, end of a string <eos>, padding <pad>, character repetition and word repetition representing tokens, etc.

### 5.3.2 Sub-word level

Sub-word tokenization means dividing a word into parts and using each part as a token. There is no specialized sub-word tokenizer for Bengali language, despite having a considerable amount of inflection. To tackle this problem, sub-word tokenization was performed using SentencePiece tokenizer [40]. SentencePiece is a language independent tokenizer that tokenizes raw sentences in an unsupervised manner. Using SentencePiece, the unigram segmentation algorithm [41] was employed to create the subword vocabulary. For training, we chose 30,000 most frequent tokens [42]. Tokens appearing less than 3 times were discarded and replaced with <unk> token. Besides, other tokens include <s> as start and </s> as the end of a sentence.

### 5.3.3 Character level

In character level tokenization, a sentence is split into characters and each character is considered a token. In this case, the total number of tokens is very low, 188 in our case by combining Bengali, English alpha-numerals and special characters. The language model generates one character at a time, which are concatenated to form words, sentences, and even paragraphs. Besides the vocabulary, we also use special tokens just as was used in word-level tokenization. The use of character level tokenization not only reduces the vocabulary size drastically but also removes the bottleneck for out-of-vocabulary words, misspellings, etc. Characters can be used to build correlation among groups of characters despite the inflection and declension that occur to add additional meaning to the word. Related words can be kept as is, without discarding any part and losing any information and still be recognized as being related.

## 5.4 Training

The main approach to perform transfer learning with the help of language models closely follows ULMFiT [9]. The entire procedure is divided into three steps, as shown in Figure 5.3. Each step has been explained in the following subsections.

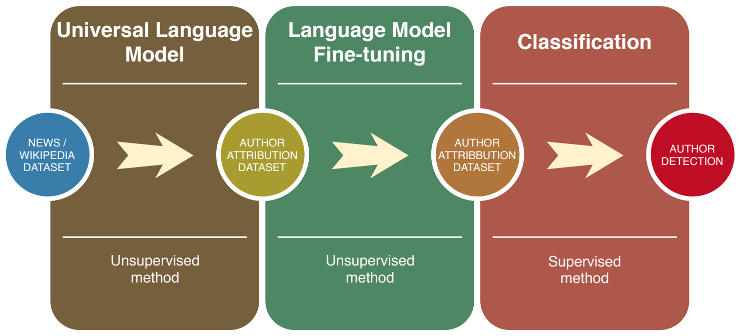


Figure 5.3: Universal Language Model Fine Tuning Steps.

### 5.4.1 Universal Language model

Two reasonably large Bengali datasets have been used to perform this step of the training. The aim for the model is to 'learn' Bengali language through language modeling. For this, we have used Wikipedia dataset and News dataset that contain a variety of Bengali texts that can generalize to most text that may appear in downstream tasks. For all the models batch size of 32, back-propagation-through-time(bptt) of 70 and weight decay of 0.1 for all layers is used. We have used a drop-out multiplier of 0.5 to the ratios selected in [9] for all the layers to avoid over-fitting. An appropriate learning rate for each model is selected through cyclic learning rate finder [32]. This is depicted in the Figure 2.1. Sometimes learning rates are altered as training progressed, as determined by the loss of the validation set. This is shown as comma separated learning rates in Table 5.1. Each epoch employed stochastic gradient descent with restarts (SGDR) mentioned in Section 2. All models are trained until they started to show signs of over-fitting, which is 30 epochs for the word and sub-word levels, and 15 epochs for the character level models. Important hyper-parameters are mentioned in Table 5.1 for all the models. After training, the models begin to learn Bengali. When provided with one or more initial words, the models can complete entire paragraphs quite meaningfully. Illustration of model predictions are shown in Figure 5.4 from the Wikipedia dataset trained model for different levels of tokenization.

Table 5.1: Universal Language model hyper-parameters

| Dataset | Tokenization | Epoch | Batch size | Learning rate |
|---|---|---|---|---|
| News | Word | 30 | 32 | 1e-3 |
|  | Sub-word | 30 | 32 | 1e-2,1e-3 |
|  | Character | 15 | 32 | 1e-2,1e-6,1e-2 |
| Wiki | Word | 30 | 32 | 1e-2,1e-3 |
|  | Sub-word | 30 | 32 | 1e-3 |
|  | Character | 15 | 32 | 1e-2 |

Word level:

**তিন মাস** দীর্ঘ সময় ধরে ধরে যাওয়া এক ধরণের ভয়ংকর খেলা । এটি একটি song খেলা যাকে সাধারণত সাধারণভাবে বলা হয় হাফ - অফ খেলা । এই খেলা উৎসবের সময় রাতের বেলা রাতের বেলায় দিয়ে শুরু হবে । তবে কিছু কিছু খেলায় দেখা যায় , তবে বেশিরভাগ সময়ই তা হতে পারে ।

Sub-word level:

**তিন মাস**ব্যাপী▁রাষ্ট্রীয়▁সম্মান▁পদক▁প্রদানের▁ব্যবস্থা▁করা▁হয়।▁বর্তমানে▁চিকিৎসা▁বিজ্ঞানে▁সবচেয়ে▁বেশি▁অবদানের▁জন্য▁এ▁সম্মাননা▁প্রদান▁করা▁হচ্ছে।▁১▁মে,▁২০১৭▁তারিখে▁সর্বপ্রথম▁এই▁পুরস্কার▁প্রদান▁করা▁হয়।▁xxbos▁ইরসাল-উল-হক▁(মৃত্যু:▁১৬▁মার্চ▁১৮৯৩)▁ছিলেন▁একজন▁মুসলিম▁পন্ডিত,▁লেখক,▁দার্শনিক▁ও▁রাজনীতিবিদ।

Character level:

**তিন মাস** পরে আফ্রিকা মহাদেশের বিশ্বের অন্যতম সেরা তিনটি বিষয় যেখানে আলবানীয় সম্রাট জেরুজালেমের উত্তর অভিবাসী ও সাম্রাজ্যবাদী তুর্কির নামানুসারে পরিচিত ছিল এবন উল্লেখযোগ্য রোমান বিরোধী লোকেরা এটিকে প্রভাবিত

Figure 5.4: Text generation from different language models.

### 5.4.2 Language model Fine-tuning

After training the language model on general text, it is fine-tuned to the task-specific type of text. Each model from the previous section is then fine-tuned on two authorship attribution datasets as mentioned in Section 3. In order to tune the model, the original language model is first loaded and frozen except for the last layers that hold the most task specific information. Using cyclic learning rate finder [32] learning rate for each model is determined. The model has been trained for 2 epochs, and then trained for 2 more epochs unfreezing another layer group, repeating it one more time before unfreezing the entire model. The models are trained as long as loss kept decreasing and training has been halted on signs of over-fitting. Learning rates are altered as required by observing the loss of the validation data, depicted by comma separated values in Table 5.2. Rest of the model settings are kept same as before. Batch size along with other hyper-parameters are summarized in Table 5.2.

Table 5.2: Fine-Tuning Language model hyper-parameters

| ULM Dataset | Tokenization | Dataset | Epoch | Batch size | Learning rate |
|---|---|---|---|---|---|
| News | Word | 1 | 19 | 32 | 1e-3 |
| | | 2 | 10 | 32 | 1e-2 |
| | Sub-word | 1 | 22 | 32 | 1e-3,1e-4 |
| | | 2 | 10 | 32 | 2e-2 |
| | Character | 1 | 14 | 32 | 1e-2 |
| | | 2 | 10 | 128 | 1e-3 |
| Wikipedia | Word | 1 | 26 | 32 | 1e-2,1e-3,1e-4 |
| | | 2 | 10 | 32 | 1e-2 |
| | Sub-word | 1 | 24 | 32 | 1e-2,1e-3 |
| | | 2 | 10 | 32 | 2e-2 |
| | Character | 1 | 20 | 32 | 1e-2 |
| | | 2 | 10 | 128 | 1e-1,1e-2,1e-3 |

### 5.4.3 Classification

The final down-stream task is classification. The model is modified as mentioned in Section 5.2.2. The encoder weights from the fine-tuned language model are loaded into the classifier. The decoder weights are randomly initialized. Dropout multiplier is set to 0.5 and chosen learning rate is 1e-2 using cyclical learning rate finder. We have used momentum values 0.8 and 0.7 for optimization [9]. On loading the language model weights, the model is frozen with only the decoder available for training. After training for 2 epochs, a layer group is unfrozen and trained using sliced learning rates for 2 epochs. Slicing mainly distributes learning rates among the layers so that the initial layers are updated slowly to maintain the pre-trained weights while the later layers, which are the most task specific layers, are updated swiftly to learn the task at hand. A general rule of slicing learning rate has been followed where the slice consists of initial learning rate(lr) and $\frac{lr}{2.6^4}$ [9]. The unfreezing step is carried out one more time and trained for 2 epochs before unfreezing the entire model and training for 6 epochs. Training has been stopped when it started to overfit. We have used batch size of 32 for all the models at this stage.

## 5.5 Experiments

To compare the proposed models against baselines, previous state-of-the-art models were re-created and results were drawn in terms of accuracy and F1-score. Neither of the previous models use transfer learning from entire models. They are based on using pre-trained embeddings to boost performance. The models used for comparison are discussed in brief below.

### 5.5.1 CNN-LSTM Word level classifier

The CNN-LSTM model [4] comprises of a mix of convolutional and LSTM layers suitable for training the corpus on word-level tokenization. The architecture is made up of an embedding layer that is initialized with pre-trained word-embeddings using skip-gram of fastText. Vocabulary size of 60,000 and embedding vector size of 300 are used. Embedding layer is followed by 2 convolutional and maxpool layer pairs. The outputs from the convolutional layers are fed into an LSTM layer of 100 neurons, whose outputs are in-turn fed into a fully connected layer of 512 neurons. Dropout is used in the fully connected layer to prevent the model from overfitting. A final softmax layer outputs probabilities for the classification. All parameters for the layers are kept same as the originally proposed model. The model is trained for 15 epochs with a learning rate of 0.001, decay of 1e-4 and batch size of 128 with 750 words per sample. We have used Adam optimizer and categorical cross-entropy as the loss function.

### 5.5.2 CNN Character level classifier

This model aims to use character signals as embedding and passes them on to several convolutional layers to extract information and later performs classification. The original CNN model [4] has been recreated and trained for the entire dataset. The first layer consists of the embedding layer whose weights are set as the pre-trained weights [4] obtained from training the News dataset mentioned in Section 3. The vocabulary and embedding size is 181. The vocabulary is the same as the one used with the character-level proposed model except for the special tokens. The embedding layer is connected to 4 sequentially connected convolutional layers. Each convolutional layer is followed with a maxpool layer of kernel size 3. A fully connected layer is stacked on top with the number of nodes set as 512. The number of filters and kernel sizes for the convolutional layers is kept same as the original model. The last layer is the output layer with softmax activation. With a batch size of 128 and a maximum text sample length of 3000, the model has been trained on the dataset for 15 epochs. A dropout rate of 0.5 is used in the fully connected layer as regularization. Rest of the hyper-parameters are kept consistent with the CNN-LSTM model.

## 5.6 Results and Discussions

All the models were measured in terms of accuracy and F1-score. Although Dataset2 is balanced, Dataset1 is not(refer to section 3), for which a look on the weighted F1-score is also necessary. Accuracy measures the percentage of samples correctly identified, whereas the F1 score is the harmonic sum of the precision and recall. Weighted F1-score thus gives us the ability to look into samples correctly identified from each class in a comparable form. Table 5.3 shows the summarization of the results obtained from various models against all the proposed models.

The effects of various other factors are analysed on the obtained results, such as model, tokenization, number of author, sample distribution, and dataset used for pre-training. These are

Table 5.3: Results of Classification

| **Dataset** | **Model** | **Tokenization** | **ULM Dataset** | **FiT Perplexity** | **Accuracy%** | **F1 Score** |
|---|---|---|---|---|---|---|
| Dataset1 | **ULMFiT** | Word | News | 74.67 | 99.58 | 0.9855 |
| | | | Wiki | 60.91 | 99.67 | 0.9967 |
| | | **Sub-word** | **News** | 62.45 | **99.80** | **0.9980** |
| | | | Wiki | 57.85 | 99.72 | 0.9972 |
| | | Character | News | 3.42 | 98.55 | 0.9855 |
| | | | Wiki | 3.36 | 98.58 | 0.9858 |
| | Char-CNN | Character | - | - | 86.28 | 0.7981 |
| | CNN-LSTM | Word | - | - | 93.82 | 0.8934 |
| Dataset2 | **ULMFiT** | Word | News | 203.11 | 94.67 | 0.9469 |
| | | | Wiki | 149.58 | 94.33 | 0.9437 |
| | | **Sub-word** | **News** | 233.04 | **95.33** | **0.9536** |
| | | | Wiki | 260.90 | 94.67 | 0.9476 |
| | | Character | News | 4.47 | 83.67 | 0.8360 |
| | | | Wiki | 3.71 | 90.00 | 0.9007 |
| | Char-CNN | Character | - | - | 73.77 | 0.7147 |
| | CNN-LSTM | Word | - | - | 66.33 | 0.6320 |

discussed briefly below.

### 5.6.1 Model Effect

In order to assess the effects of ULMFiT model for the task of authorship attribution we train some previous state-of-the-art models, namely Char-CNN and word level CNN-LSTM. From Table 5.3 it is evident the ULMFiT model outperforms the other models with a significant amount on both Dataset1 and Dataset2. This clearly shows that the transfer learning approach is effectively applicable to the task of authorship attribution. The model learns Bengali from the language model training and tunes to authorial writing styles on fine-tuning. Thus with the additional steps of teaching the model a language, it is able to perform better in detecting the author of a text. Besides this, the AWD-LSTM architecture along with the training techniques employed offers a strong base for the language model as well as the classifier making the transfer learning thereof highly effective [9].

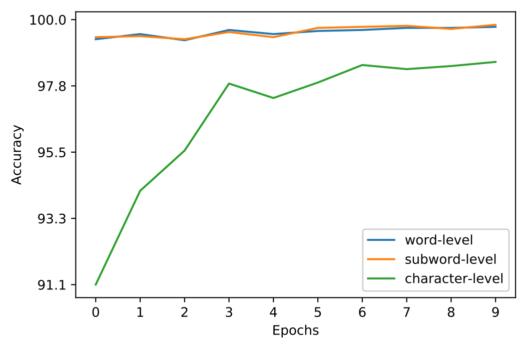
### 5.6.2 Tokenization Effect

Tokenization is an important part of any NLP task. In Bengali, the most common way so far has been to tokenize by words. In this paper three types of tokenizations are attempted and their effects are analyzed on the task of author classification. As described in Section 5.3, we have performed word, sub-word and character level tokenization. In general, the character level models perform worse than both word and sub-word models. This reflects that the character level model is facing difficulty choosing the right author because of the long stream of tokens it has to go through before reaching any conclusion. LSTM layers pass one character at a time, making any sentence a long set of tokens. The LSTM layers therefore may have trouble gathering enough data about the difference in text or style for each author.

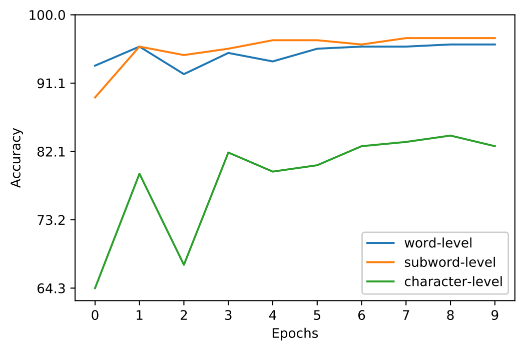
The word and sub-word level models perform nearly equally well, but on further epochs, the sub-word models begin to outrun the word-level models. These information can be drawn from Figure 5.5 where we see the progress for each tokenization type model on both Dataset1 and Dataset2 on a zoomed-in scale. The comparison of word and sub-word level models can be inferred from Figure 5.6 where we see the increasing accuracy of the sub-word level model on each epoch. The sub-word tokenization breaks down the text into multiple parts but not completely. As a result, linguistic information about the words are kept intact and it also provides information about the relationship between structural components of words. Whereas, when words are simply split, relation between words with slight variations of the structure is not as easily recognized.

### 5.6.3 Effect of number of Authors

In this section, the proposed models with varying numbers of authors are compared to determine the effectiveness of the models against increasing number of classes with fewer samples to train on. Dataset1 contains 16 authors in an imbalanced manner. It is chunked into 5 different parts containing 6, 8, 10, 12 and 14 authors respectively taking subsets randomly from the original dataset. The samples per class in the derived datasets are truncated to the minimum number of samples from among the classes. We have only compared the accuracy because all the models are now being trained and tested on balanced dataset. Accuracy is measured on 20% held out dataset for each case, after training on 80% of the data. Each of the 6 proposed models of 3 types(word, sub-word and character level) are trained on these 5 sub-datasets. It can be inferred that as we

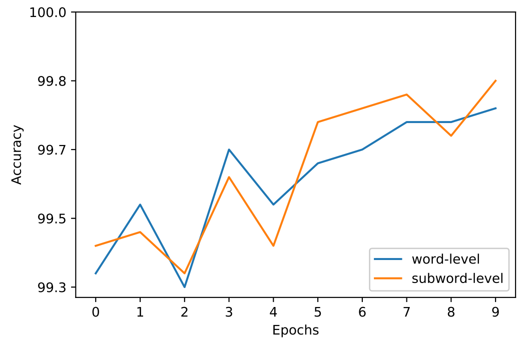


(a) On Dataset1

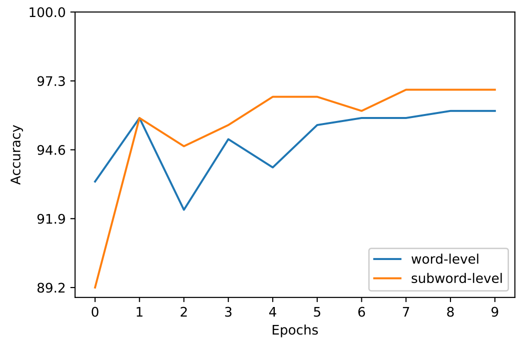


(b) On Dataset2

Figure 5.5: Effect of tokenization on classification



(a) On Dataset1



(b) On Dataset2

Figure 5.6: Effect of word and sub-word tokenization on classification

increase the number of authors, the sample per author also decreases, making it difficult for any deep learning model to learn each class of text. The summary is presented in Table 5.4. A graph depicting the accuracy trend with an increase in the number of authors for all the models is shown in Figure 5.7.

Out of the 5 subsets that are created, the sub-word model pre-trained on the Wikipedia corpus performs best in majority cases (3 in this occasion). More importantly, as the sample number decreases, all other models more or less start performing worse than the Wikipedia sub-word model. Figure 5.7 shows the decline in character model with lesser samples to train on. Moreover, the word and sub-word level wiki models consistently perform better and the subword model tends to give higher accuracy. From these results, we can conclude that the Wikipedia sub-word model shows more stability than others. The reason behind this could be two-fold. Firstly the Wikipedia

Table 5.4: Accuracy for various number of Authors

| | | # of authors on the subset | | | | |
|---|---|---|---|---|---|---|
| Tokenization | ULM Dataset | 6 authors | 8 authors | 10 authors | 12 authors | 14 authors |
| Word | News | **99.69** | 99.53 | 99.58 | 99.41 | 98.63 |
| | Wiki | 99.62 | 99.56 | **99.70** | 99.48 | 99.47 |
| Sub-word | News | 99.47 | 99.40 | 98.71 | 99.56 | 99.39 |
| | Wiki | 99.62 | **99.67** | 99.41 | **99.63** | **99.54** |
| Character | News | 98.79 | 98.60 | 98.00 | 98.38 | 97.26 |
| | Wiki | 98.79 | 98.66 | 98.18 | 97.78 | 97.11 |

dataset has more varied text than News dataset. This helps the model generalize better. Secondly, the sub-word level model provides enough information about word and sentence structure yet not breaking it down too much(into characters) to lose sight of the overall picture of the topic being sampled.

### 5.6.4 Effect of pre-training datasets

Although Wikipedia trained models tend to show better down-stream generalization, the scenario changes in real-world applications. Data is often imbalanced with some authors having very few training samples compared to others. The loss for each epoch is plotted to compare the effects of News and Wikipedia pre-trained models for each dataset(1 and 2) in Figure 5.8. For both the datasets, the news-trained language model when used for transfer learning performs better in terms of loss. The loss tends to reduce steadily with each epoch on both cases, but for news-model, the loss gets much lower than the wiki-model. This shows that the categories and size of News dataset enabled the classifier to distinguish better among the authors.

## 5.7 Pre-trained Language Models

A set of language models have been pre-trained on two different datasets, namely News and Wikipedia as part of our workflow. Language models are measured in terms of perplexity. This measure captures the degree of uncertainty in predicting the next word of the sentence. It is calculated as the exponentiation of the obtained loss. Low perplexity is a sign of a well-trained model. Table 5.5 lists the perplexities of the pre-trained models. We see that the character-level models perform significantly better for the task of language modeling. An illustration of the

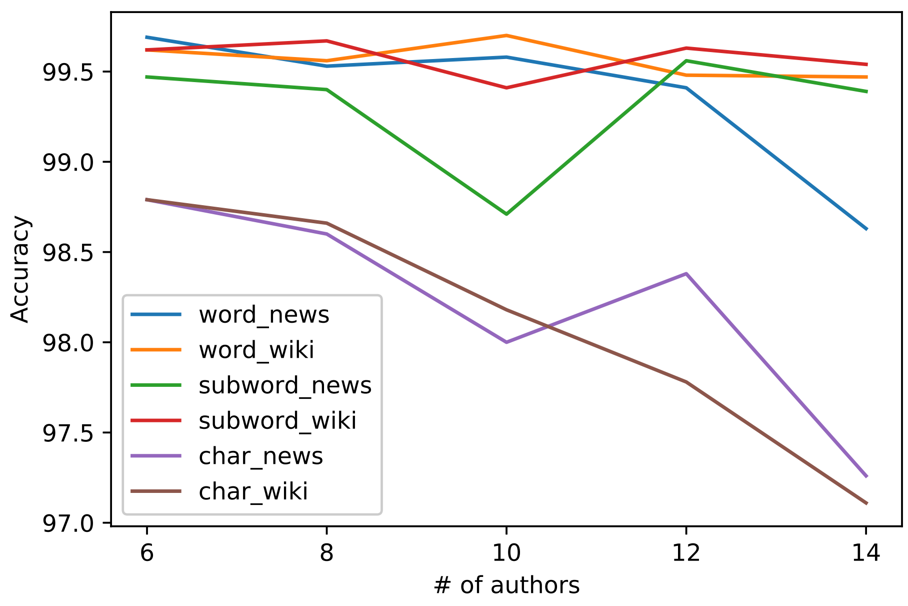
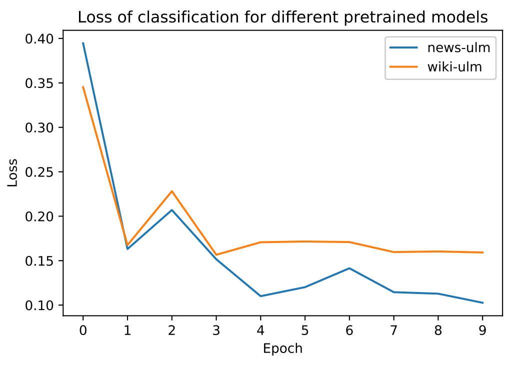


Figure 5.7: Performance of different models with increasing number of authors

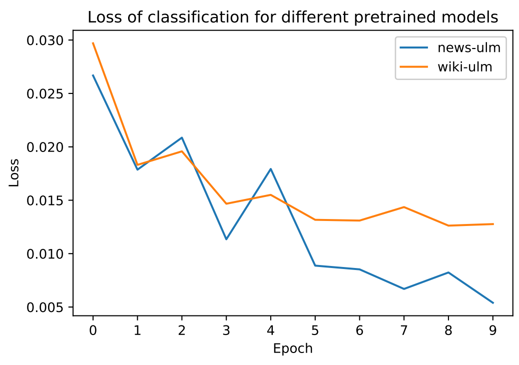
produced text can be seen in Figure 5.4. The next best performance is obtained by the sub-word level model being slightly lower than the word-level models. The reason for this can be inferred that the representation of the tokens in smaller forms(e.g. characters) may provide additional information about the words and relation among the words than larger chunks can provide.

Table 5.5: Perplexity of Universal Language Models

| **Dataset** | **Tokenization** | **Perplexity** |
|---|---|---|
| News | Word | 47.52 |
| | Sub-word | 43.61 |
| | Character | 3.20 |
| Wiki | Word | 83.62 |
| | Sub-word | 74.20 |
| | Character | 3.08 |



(a) On Dataset1



(b) On Dataset2

Figure 5.8: Effect of dataset used in pre-training on classification

# Chapter 6

# Phase 3: Transformer Based Transfer Learning for Authorship Attribution

## 6.1 Overview

Text classification is a classical natural language processing task. An important intermediate step for this task includes text representation. Recent developments in this field has introduced and developed on the concept of transfer learning, specifically various pre-trained models are being trained on large corpuses and being released for public use. These models serve as a starting point for various tasks instead of training random models from scratch. So far we have used and analysed word embeddings, character embeddings and LSTM based language modeling objectives. Since the beginning of 2019, a new type of architecture has come to light and has surpassed most previous records on various NLP tasks. They are transformer architectures [43]. Among the various models that use this architecture are BERT [44], XLNet [45], OpenAI GPT [46], RoBERTa [47] and multilingual models such as mBERT, XLM [48] and XLM-R [49]. The common factor in all these models is 1. They are all large models 2. They have been pretrained on vary large amount of data and can even be trained more. Most models though trained on primarily single languages such as English, German and Finnish some multilingual models are also available. mBERT for example is trained on the wikipedias of over 100 languages, XLM on 15 and XLM-R on common crawl corpus of over 100 languages. Due to heavy computational and large coupus requirement no such model has been pretrained solely on Bengali yet. Nevertheless we leverage the multilingual model mBERT which contains a small fraction of Bengali text on its pretraining data. In this phase of our research we use the transformer based multilingual model mBERT to perform Authorship Attribution on Bengali texts. We also compare its performance with our previous approaches and shed light into the potential improvements in our methodologies.

## 6.2 Architecture

BERT's architecture is a multi-layer bidirectional Transformer encoder based on the original Transformer [43]. The uniqueness of BERT is in that it uses bidirectional context instead of only left-right context as in casual language modeling task. This new kind of language modeling is called masked language modeling, similar to what is commonly known as 'cloze' task. Besides language modeling, BERT also employs next sentence prediction task when pretraining. The architecture is formed of two parts: a transformer base, and task specific head. A brief discussion of these parts is given below.

### 6.2.1 Input/Output Representation

A Transformer is usually made of two parts, and encoder and a decoder and is usulaly used for seq2seq tasks. BERT on the other hand uses only the encoder part of the transformer model. BERT can handle a variety of tasks including Question Answering, text classification etc. BERT uses WordPiece embeddings [50] where a word maybe be broken into multiple pieces and each piece is prefixed with the symbols ## to indicate their origins. The first token of all sequence is a special token [CLS]. The final hidden state of this token is used as an aggregate sequence representation. Besides this, each input sequence is a pair of sequences separated by the token [SEP]. BERT has 3 kinds of embeddings. Word embedding, positional embedding and sentence embedding. Word embeddings map each word to a vector space, where word is a WordPiece token. Because this model is masked language model, it cannot implicitly get information about the sequence in which the words are. To tackle this positional embedding is used. Sentence embedding represents if a sequence precedes another sequence in the next sentence prediction task of the model. The final embedding of the model is therefore the sum of these three embeddings.

### 6.2.2 Transformer Base and pretraining

The base architecture of BERT is exactly like the encoder of the original transformer model. We use the smaller BERT model called the BERT base. It has L = 12 attention blocks, A = 12 self-attention heads and the hidden size is H = 768 [44]. Since it does not have the decoder part, it also does not have any masked attention or encoder-decoder attention. Towards the end of the model, all the attention heads are merged and a single output representation is presented. For the purposes of classification we only concern ourselves with the output of the [CLS] token present at the beginning of every sentence.

BERT is pretrained with two tasks. Masked Language Modeling and next sentence prediction task. Masked language model tries to predict the masked token. In the experiments, they mask 15% of the tokens randomly and calculate the loss based on their predictions. To tackle the downstream tasks that require understanding the relationship between sentences such as Question Answering and Natural Language Inference, the task of next sentence prediction was introduced. This task predicts if B is the sentence next to A or not in the sequence $[CLS]A[SEP]B$. The combination of these two tasks make the model more robust towards downstream tasks.

### 6.2.3 Classification

For classification of the authors, we add a single layer classifier on top of the BERT transformer encoder. Specifically the final hidden vector $C \in \mathbb{R}^{\mathrm{H}}$ corresponding the input token [CLS] is input to the classifier layer. And the number of outputs is the number of class labels, in our case the number of authors in the dataset. Then the entire model is then fine tuned along with the new layer end-to-end.

For our task of authorship attribution in Bengali we use the trained multilingual BERT model called *bert-base-multilingual-cased*. We use a batch size of 6, total sequence length of 512 as per the original setting. Most of the settings are set same as the original model as well. Unlike original BERT we use Slanted Triangular learning rates, Discriminative Learning rates and gradual unfreezing similar to our previous approach. We select the learning rates at various stages using learning rate finder. Thus we select an initial learning rate of 0.0004 for Dataset1 and train the classifier for 40 epochs. For Dataset2 learning rate 0.0008 was selected and was trained for 35 epochs. The number of epochs were selected based on the validation set errors. When the validation set errors start to increase or oscillate, the training is stopped.

## 6.3 Results and Discussion

The models trained were measured in terms of their accuracy and then compared with our previous approaches. Table 6.1 shows the comparison for both the datasets.

Table 6.1: Results of Classification

| Dataset | Model | Accuracy% |
|---|---|---|
| Dataset1 | mBERT | 94.79 |
| | ULMFiT-subword-news | **99.80** |
| | Char-CNN | 86.28 |
| | CNN-LSTM | 93.82 |
| Dataset2 | mBERT | 93.33 |
| | ULMFiT-subword-news | **95.33** |
| | Char-CNN | 73.77 |
| | CNN-LSTM | 66.33 |

mBERT was preatrained on over 100 languages wikipedia text, among which one of the languages is Bengali. The wikidump of Bengali is not of a significant size as apparent from experimentation's in our previous ULMFiT based approach. Yet when finetuned on authorial texts the mBERT models achieves accuracies near the previous best performances. This shows the potential of the BERT architecture and its pretraining. Our previous approach on the other hand used news text corpus which contained much more text data than the wikipedia dump. Since the BERT model was not trained solely on Bengali we hypothesize a few further improvements that may make BERT the state-of-the-art in authorship attribution in Bengali as well.

## 6.4 Further Works

Some of the future improvements that we assume will use BERTs full potential are:

- Finetuning the mBERT model on language modeling task on only Bengali text. For this purpose we need a larger Bengali text corpus and lots of compute time. On further pretraining we assume the model will be able to work with pure Bengali texts with much higher accuracy.

- Creating a Bengali BERT model. This is a major undertaking and will require a very large Bengali corpus and even larger amount of time. Although a lot of work, if accomplished, this pretrained Bengali model will very beneficial to not only authorship attribution, but most other NLP tasks in Bengali including classification, Reasoning and Question Answering.

# Chapter 7

# Conclusion And Future Works

We start authorship attribution exploration through character-level model and show their competitive performace but superior memory and time considerations. We also pretrain character embeddings and perform comparative analysis with word based models.

These models require large amounts of data to perform better and so we move on to transfer learning through language modeling objective. We use a multi-layer LSTM based network as a language learner and use it for downstream tasks, such as authorship attribution. Overall, the approach is comprised of three chronological phases with the first phase being the training of a language model with generic text, the second phase being the fine-tuning of the same model on unlabeled authorial texts and the final phase being a classifier trained to detect the authors. Finally, for comparative analysis, we train various previously proposed models on two datasets containing literary works in Bangla and see that this approach outperforms other traditional methods in both the cases. More importantly however, this method seems to provide more stability with fewer data and a larger pool of authors to detect from. Furthermore, various levels of tokenizations were performed and the results were compared to discover that the Wikipedia based sub-word tokenized model performs better consistently even with a small number of samples to learn from. The three trained language models with different tokenizations along with the codes have been released.

Finally we start basic exploration of transformer based models for transfer learning and find promising results. We propose some future paths to use BERTs full potential.

We have three papers published and 1 journal paper under review for publication. We would kindly request to avoid direct plagiarism of any portion of this report. For any new idea inspired from this report citations will be appreciated.

## 7.1 Future Plan

We have just scratched the surface of the domain of transfer learning and a number of different research opportunities have risen from this. The possible future path for this research includes:

- **More Transformer based transfer learning**
- **Character-level language model and transfer learning thereof**
- **Cross-lingual authorship attribution**

# Chapter 8

# Publications

On various stages and components of our research we have published a few papers. They are mentioned below:

- **Authorship Attribution in Bangla literature using Character-level CNN.** Published in ICCIT, 2019.
- **A Continuous Space Neural Language Model for Bengali Language.** Published in ICCIT, 2019.
- **A Subword Level Language Model for Bangla Language.** Published in IJCCI, 2019.
- **A Transfer Learning approach on Authorship Attribution in Bangla Literature.** Under review in Journal.

# References

[1] S. Lynn-Evans. (2019) Ten techniques learned from fast.ai. [Online]. Available: https://blog.floydhub.com/ten-techniques-from-fast-ai/

[2] C. Zhang, X. Wu, Z. Niu, and W. Ding, "Authorship identification from unstructured texts," *Knowledge-Based Systems*, vol. 66, pp. 99–111, 2014.

[3] E. Stamatatos, "A survey of modern authorship attribution methods," *Journal of the American Society for information Science and Technology*, 2009.

[4] M. S. I. M.-E.-J. Aisha Khatun, Anisur Rahman, "Authorship attribution in bangla literature using character-level cnn," *International Conference on Computer and Information Technology (ICCIT)*, 2019.

[5] F. Jafariakinabad, S. Tarnpradab, and K. A. Hua, "Syntactic recurrent neural network for authorship attribution," *arXiv preprint arXiv:1902.09723*, 2019.

[6] S. Ruder, P. Ghaffari, and J. G. Breslin, "Character-level and multi-channel convolutional neural networks for large-scale authorship attribution," *arXiv preprint arXiv:1609.06686*, 2016.

[7] P. Das, R. Tasmim, and S. Ismail, "An experimental study of stylometry in bangla literature," 2015.

[8] H. A. Chowdhury, M. A. H. Imon, and M. S. Islam, "A comparative analysis of word embedding representations in authorship attribution of bengali literature," 2018.

[9] J. Howard and S. Ruder, “Universal language model fine-tuning for text classification,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 328–339.

[10] T. Kreutz and W. Daelemans, “Exploring classifier combinations for language variety identification,” in *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects, Santa Fe, New Mexico, USA, August 20, 2018*, 2018, pp. 191–198.

[11] K. Sundararajan and D. Woodard, “What represents "style" in authorship attribution?” in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 2814–2822.

[12] D. Bagnall, “Authorship clustering using multi-headed recurrent neural networks-notebook for pan at clef 2016,” in *CLEF 2016 Evaluation Labs and Workshop–Working Notes Papers*, 2016, pp. 5–8.

[13] P. Shrestha, S. Sierra, F. Gonzalez, M. Montes, P. Rosso, and T. Solorio, “Convolutional neural networks for authorship attribution of short texts,” in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 2017, pp. 669–674.

[14] J. Hitschler, E. van den Berg, and I. Rehbein, “Authorship attribution with convolutional neural networks and pos-eliding,” in *Proceedings of the Workshop on Stylistic Variation*, 2017, pp. 53–58.

[15] O. Fourkioti, S. Symeonidis, and A. Arampatzis, “Language models and fusion for authorship attribution,” *Information Processing & Management*, vol. 56, no. 6, p. 102061, 2019.

[16] M. T. Hossain, M. M. Rahman, S. Ismail, and M. S. Islam, “A stylometric analysis on bengali literature for authorship attribution,” 2017.

[17] U. Pal, A. S. Nipu, and S. Ismail, “A machine learning approach for stylometric analysis of bangla literature,” 2017.

[18] S. Phani, S. Lahiri, and A. Biswas, “A machine learning approach for authorship attribution for bengali blogs,” 2016.

[19] I. Santos, N. Nedjah, and L. de Macedo Mourelle, “Sentiment analysis using convolutional neural network with fasttext embeddings,” 2017.

[20] E. Rudkowsky, M. Haselmayer, M. Wastian, M. Jenny, Š. Emrich, and M. Sedlmair, “More than bags of words: Sentiment analysis with word embeddings,” *Communication Methods and Measures*, 2018.

[21] K. S. Tai, R. Socher, and C. D. Manning, “Improved semantic representations from tree-structured long short-term memory networks,” *arXiv preprint arXiv:1503.00075*, 2015.

[22] H. He, K. Gimpel, and J. Lin, “Multi-perspective sentence similarity modeling with convolutional neural networks,” 2015.

[23] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu, “Charagram: Embedding words and sentences via character n-grams,” *arXiv preprint arXiv:1607.02789*, 2016.

[24] W. Ling, T. Luís, L. Marujo, R. F. Astudillo, S. Amir, C. Dyer, A. W. Black, and I. Trancoso, “Finding function in form: Compositional character models for open vocabulary word representation,” *arXiv preprint arXiv:1508.02096*, 2015.

[25] M. Ballesteros, C. Dyer, and N. A. Smith, “Improved transition-based parsing by modeling characters instead of words with lstms,” *arXiv preprint arXiv:1508.00657*, 2015.

[26] M.-T. Luong and C. D. Manning, “Achieving open vocabulary neural machine translation with hybrid word-character models,” *arXiv preprint arXiv:1604.00788*, 2016.

[27] J. Chung, K. Cho, and Y. Bengio, “A character-level decoder without explicit segmentation for neural machine translation,” *arXiv preprint arXiv:1603.06147*, 2016.

[28] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” 2015.

[29] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, “Exploring the limits of language modeling,” *arXiv preprint arXiv:1602.02410*, 2016.

[30] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” in *Proceedings of the 2018 Conference of the*

*North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 2227–2237.

[31] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," in *Advances in neural information processing systems*, 2015, pp. 3079–3087.

[32] L. N. Smith, "No more pesky learning rate guessing games," *CoRR*, vol. abs/1506.01186, 2015. [Online]. Available: http://arxiv.org/abs/1506.01186

[33] I. Loshchilov and F. Hutter, "SGDR: stochastic gradient descent with restarts," *CoRR*, vol. abs/1608.03983, 2016.

[34] A. Radford, R. Jozefowicz, and I. Sutskever, "Learning to generate reviews and discovering sentiment," *arXiv preprint arXiv:1704.01444*, 2017.

[35] K. Cao and M. Rei, "A joint model for word embedding and word morphology," *arXiv preprint arXiv:1606.02601*, 2016.

[36] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing LSTM language models," *CoRR*, vol. abs/1708.02182, 2017.

[37] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," *Advances in neural information processing systems*, 2016.

[38] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing lstm language models," *CoRR*, 2018.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[40] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018.

[41] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," *arXiv preprint arXiv:1804.10959*, 2018.

[42] P. Czapla, S. Gugger, J. Howard, and M. Kardas, "Universal language model fine-tuning for polish hate speech detection," *Proceedings ofthePolEval2019Workshop*, p. 149, 2019.

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available: https://arxiv.org/pdf/1706.03762.pdf

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

[45] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in neural information processing systems*, 2019, pp. 5754–5764.

[46] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

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

[48] G. Lample and A. Conneau, "Cross-lingual language model pretraining," *arXiv preprint arXiv:1901.07291*, 2019.

[49] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," *arXiv preprint arXiv:1911.02116*, 2019.

[50] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.