INDIAN STATISTICAL INSTITUTE, KOLKATA

# "AUTHOR IDENTIFICATION AND ANALYSIS OF BOLLYWOOD SONG LYRICS"

A DISSERTATION SUBMITTED BY
DEEPESH PAI

In the partial fulfillment of the requirements for the degree of
Master of Technology in Computer Science

UNDER THE GUIDANCE OF
DEBAPRIYO MAJUMDAR

**MASTERS OF TECHNOLOGY IN COMPUTER SCIENCE**
INDIAN STATISTICAL INSTITUTE
205, B.T Road, Kolkata 700108

## CERTIFICATE

I hereby certify that the Project Dissertation titled "**Author Identification And Analysis Of Bollywood Song Lyrics**" which is submitted by **Deepesh Pai (CS1920)**, **M.Tech. in Computer Science**, Indian Statistical Institute, Kolkata in partial fulfillment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Kolkata

Date: 09.07.2021

Supervisor

**MASTERS OF TECHNOLOGY IN COMPUTER SCIENCE**
INDIAN STATISTICAL INSTITUTE
205, B.T Road, Kolkata 700108

## CANDIDATE'S DECLARATION

I, **Deepesh Pai (CS1920)**, students of M.Tech in Computer Science, hereby declare that the project Dissertation titled "**Author Identification And Analysis Of Bollywood Song Lyrics**" which is submitted by me to the **M.Tech. (CS) Dissertation Committee 2020-21, ISI Kolkata** in partial fulfillment of the requirement for the award of the degree of Masters of Technology, is original and not copied from any source without proper citation.

| | |
|---|---|
| Place: Kolkata | Deepesh Pai |
| Date: 09.07.2021 | CS1920 |

# Acknowledgment

I would like to express my sincere gratitude to my supervisor Prof. Debapriyo Majumdar for giving me an opportunity to work under him and providing me exciting problems to think upon. Throughout the development of this project, he was always available for discussion and guidance. I would also like to thank Barkha Bharti for providing me useful suggestions and resources in creating the dataset using web crawling. Finally, I would like to express my special gratitude to my first teacher my parents, who have always encouraged me in all aspects, for igniting the love for knowledge within me.

# Abstract

Research in Natural Language Processing is expanding in multiple domains and applications. With every advancement, the variety of text that are processed is growing. One such domain is lyrics processing.
Songs are vital to the music and film industry and are analyzed to get important information such as genre, theme, mood, author, etc. of the song. Bollywood, the Indian film industry makes a lot of revenue making use of songs. The number of songs churned out by this industry is massive and is a rich source of textual data for Natural Language Processing tasks. In the field of Natural Language Processing (NLP) one of the important topics is Authorship identification.
Authorship identification is the task of identifying the author of a given text from a set of authors. Authorship identification is applied to tasks such as identifying anonymous authors, detecting plagiarism, or finding ghostwriters. It also gives us an opportunity to work on data in Devanagari which is a relatively less explored field.

The main concern of this task is to define an appropriate characterization of texts that captures the writing style of authors. Although deep learning is used in different author identification tasks using LSTM and GRU, it has not been used with BERT(to the best of our knowledge). In this study, the project aims to build a system that can identify the lyricist of a song based on its lyrics. We have built a model based on BERT which would take input the lyrics of a particular song and our program would predict its lyricist based on the content of the lyrics. The results show that the proposed system outperforms its counterparts.

# Contents

# Chapter 1

# Introduction

A published author usually has a unique writing style in his works. The writing style is to some extent irrelevant to the topics and can be recognized by human readers. The process of identifying the author from a group of candidates according to his writing samples (sentences, paragraphs, or short articles) is authorship identification. The importance of authorship identification lies in its wide applications. It can be used to find the original author of widely reprinted articles, as well as plagiarised paragraphs. It provides a new way to recommend to a reader the authors who have a high similarity of writing style with his/her favorite authors or anonymous writers, etc. In our work, two labeled datasets are adopted to train and test our models. Several most advanced deep learning models are utilized at different levels to evaluate the accuracy of authorship identification.

## 1.1 Motivation

Songs are an integral part of our culture and film industry and can be analyzed to obtain important information such as the genre, theme, mood, etc. of the song. The number of songs churned out by this Bollywood is massive and is a rich source of text data for NLP tasks. One of the most important parts of the song's content is the lyrics, in which lyricists expose feelings or thoughts that may reflect their way of seeing the world. For songs, lyrics text is one of the key factors that make listeners feel songs are attractive because it delivers messages and expresses emotion. According to IFPI, More than 85% of online music subscribers search for song lyrics, indicating a keen interest of people in song lyrics. It also allows us to work on data in Hindi which is a relatively less explored field. Hindi is the most widely spoken language in India, almost 41% of the Indian population being its native speakers. It is also one of the Indian languages with a good repertoire of tools and resources for Natural Language Processing research. Hence it was interesting for us to foray into this field and work on Hindi data available in Devanagari script and combine it with research topics such as author identification. As no corpus existed for Bollywood lyrics, a corpus with 15000 song lyrics with lyricists was created with lyrics in the

Devanagari script. According to Forbes, Music from Bollywood constitutes almost 80% of music sales in India. This is a huge market and it would be very critical to work on the better organization of this data. People choose to listen to different music based on their mood and situations. Researchers have been successful in generating and analyzing lyrics for songs in English and Chinese. But there is no work (to the best of our knowledge) that explores the Hindi lyrics dataset.
Given the popularity of Hindi songs across the world and the ambiguous nature of Hindi script, The goal of this research is, Testing whether the deep learning approach can successfully predict lyricists based solely on lyrics.

## 1.2 Challenges

One of the biggest challenges in any research approach is to find the right data, both in terms of quality and quantity. A part of this thesis aims at solving this hurdle for the given domain and language. The creation of this data resource involved several tricky tasks that were to be solved by us. This involved availability of Hindi data predominantly transliterated in the Roman script. This was not suitable as the transliteration does not follow conventions and also there are some English words with mean something else in Hindi words that are transliterated such as 'man' (mind in Hindi) or 'hum' (we in Hindi). Also, the inconsistency results in a lot of errors when available transliteration tools are used. We did not want to focus our efforts on the transliteration task and hence decided to crawl the data available in Devanagari. A corpus with 15000 song lyrics with lyricists was created with lyrics in the Devanagari script. Another challenge that we faced was Lyrics data is dissimilar from other regular text as it does not follow strict grammar. To incorporate rhythm and 'matra', grammar is sometimes given a miss. Also, the amount of textual content available in lyrics is lesser than a novel or an article or a research paper, etc.

## 1.3 Contribution

Any research work is of little significance if it does not have novel contributions to the research community. The contributions of this thesis work are:
Resources and datasets are a major pillar for any sphere of research. Annotated datasets help validate research work and publications and also are vital to gaining insights to form hypotheses and understanding indications towards how to solve a given problem. A major contribution of this work is an annotated dataset of Bollywood song lyrics in the Devanagari script.
Pretraining BERT model on a domain-specific corpus. The word embedding extracted from this model will help to perform a different task on lyrics data like bias detection etc.

# Chapter 2

# Related Work

Authorship identification is not a research area that emerged out of the increased use of the internet. It was used for determining which author wrote a chapter or page of a book. Authorship identification research makes use of the structure of a text and the words that are used. A subdivision of this is stylometric research in which linguistic characteristics are used to identify the author of a text. Most of the features used for authorship identification are stylometric, especially in literary authorship. In stylometry research, it is generally accepted that authors have unconscious writing habits. These habits become evident in for example their use of words and grammar. The more unconscious a process is, the less controllable it is. Therefore, words and grammar could be a reliable indication of the author. These individual differences in the use of language are referred to as idiolect. The unconscious use of syntax gives rise to the opportunity to perform author identification based on stylometric features.

Author identification is by far the most prevalent field of authorship analysis in terms of published studies. Authorship identification studies advance rapidly in recent years, with the development of scientific research such as information retrieval and deep learning, as well as the well-organized text data sets on the Internet. The most recent studies of authorship identification are primarily based on machine learning approaches, such as deep neuron networks, and succeed with the application of various vector representations of words.

## 2.1 Deep Learning-based Authorship Identification

In the work by [Narayanan, 2020] the authorship identification is performed on two news dataset (Reuters 50 50). Gated Recurrent Unit (GRU) network, and Long Short Term Memory (LSTM) network are implemented, tuned and evaluated on the performance of authorship identification. In total three models are implemented: sentence-level GRU, article-level GRU, article-level LSTM.

### 2.1.1 Preface

Recurrent Neural Networks suffer from short-term memory. If a sequence is long enough, they'll have a hard time carrying information from earlier time steps to later ones. So if we are trying to process a paragraph of text to do predictions, RNN's may leave out important information from the beginning. During backpropagation, recurrent neural networks suffer from the vanishing gradient problem. Gradients are values used to update a neural network's weights. The vanishing gradient problem is when the gradient shrinks as it back propagates through time. If a gradient value becomes extremely small, it doesn't contribute too much learning. LSTM 's and GRU's were created as the solution to short-term memory. They have internal mechanisms called gates that can regulate the flow of information. These gates can learn which data in a sequence is important to keep or throw away. By doing that, it can pass relevant information down the long chain of sequences to make predictions. Almost all state-of-the-art results based on recurrent neural networks are achieved with these two networks.

### 2.1.2 LSTM

An LSTM has a similar control flow as a recurrent neural network. It processes data passing on information as it propagates forward. The differences are the operations within the LSTM's cells. The core concept of LSTMs is the cell state and it's various gates. The cell state act as a transport highway that transfers relative information down the sequence chain. You can think of it as the "memory" of the network. The cell state, in theory, can carry relevant information throughout the processing of the sequence. So even information from the earlier time steps can make its way to later time steps, reducing the effects of short-term memory. As the cell state goes on its journey, information get's added or removed to the cell state via gates. The gates are different neural networks that decide which information is allowed on the cell state. The gates can learn what information is relevant to keep or forget during training. So we have three different gates that regulate information flow in an LSTM cell. A forget gate, input gate, and output gate. An architecture of LSTM is shown in Figure 2.1

#### Forget Gate

First, we have the forget gate. This gate decides what information should be thrown away or kept. Information from the previous hidden state and information from the current input is passed through the sigmoid function. Values come out between 0 and 1. The closer to 0 means to forget, and the closer to 1 means to keep.

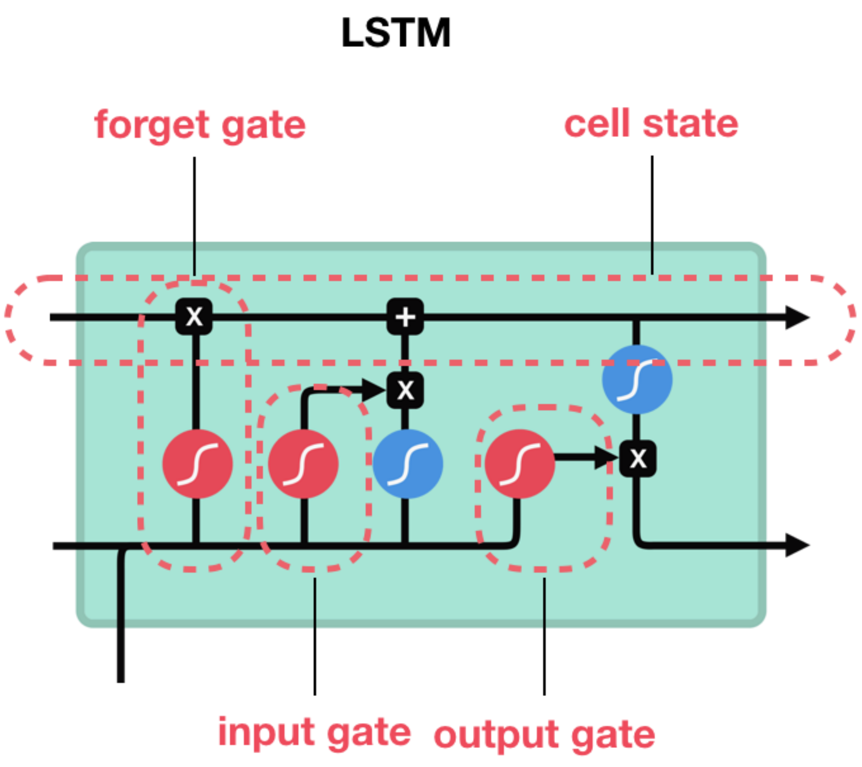


**Figure 2.1:** LSTM

**Input Gate**

To update the cell state, we have the input gate. First, we pass the previous hidden state and current input into a sigmoid function. That decides which values will be updated by transforming the values to be between 0 and 1. 0 means not important, and 1 means important. You also pass the hidden state and current input into the tanh function to squish values between -1 and 1 to help regulate the network. Then you multiply the tanh output with the sigmoid output. The sigmoid output will decide which information is important to keep from the tanh output.

**Output Gate**

Last we have the output gate. The output gate decides what the next hidden state should be. Remember that the hidden state contains information on previous inputs. The hidden state is also used for predictions. First, we pass the previous hidden state and the current input into a sigmoid function. Then we pass the newly modified cell state to the tanh function. We multiply the tanh output with the sigmoid output to decide what information the hidden state should carry. The output is the hidden state. The new cell state and the new hidden are then carried over to the next time step.

### 2.1.3 GRU

The GRU is the newer generation of Recurrent Neural networks and is pretty similar to an LSTM. GRU's got rid of the cell state and used the hidden state to transfer information. It also only has two gates, a reset gate, and an update gate. An architecture of GRU is shown in Figure 2.2

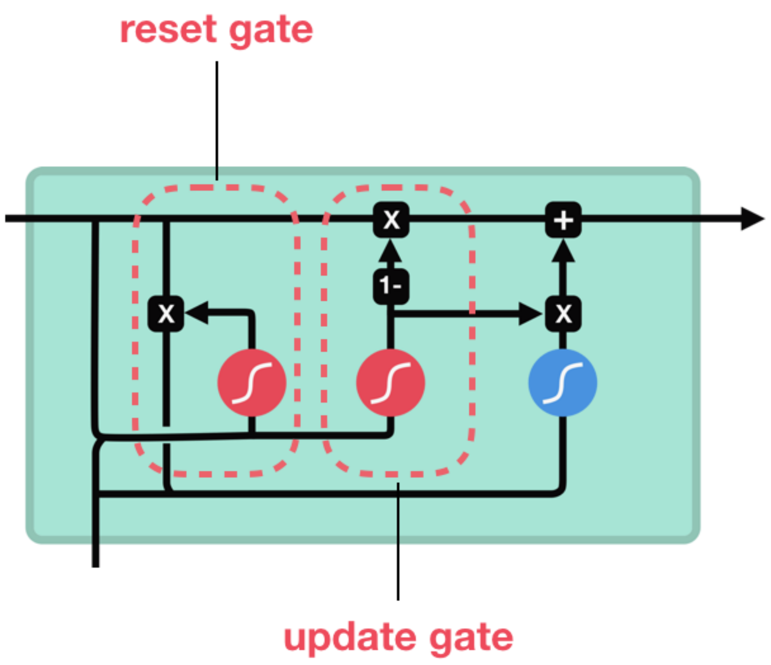


**Figure 2.2:** GRU

#### Update Gate

The update gate acts similar to the forget and input gate of an LSTM. It decides what information to throw away and what new information to add.

#### Reset Gate

The reset gate is another gate that is used to decide how much past information to forget. And that's a GRU. GRU's has fewer tensor operations therefore, they are a little speedier to train than LSTM's. There isn't a clear winner which one is better. Researchers and engineers usually try both to determine which one works better for their use case.

### 2.1.4 Sentence-Level GRU

In this model, the input unit is a word. For each timestamp, one word, represented as a word vector is sent to the GRU network, updating the hidden state and producing some output. After the last word in the sentence bundle is processed, all outputs at the previous timestamp are sent to a pooling layer, whose output is simply the average of all inputs. Finally, the output of the pooling layer will be used as the input of a softmax classifier, where the loss will be calculated and the final decision that which author wins out will be determined. The model is shown in FIGURE 2.3.

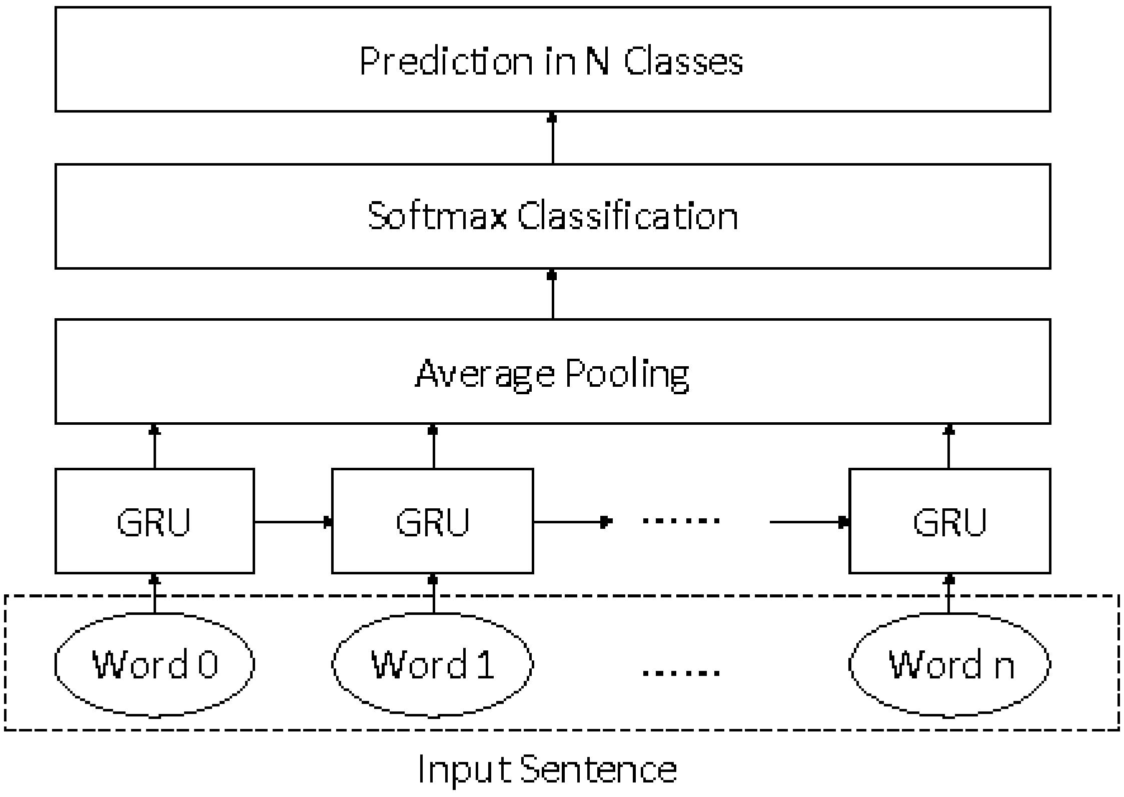


**Figure 2.3:** Sentence Level GRU Diagram

### 2.1.5 Article-Level GRU

Different from sentence-level GRU, article-level GRU takes a whole or more paragraphs as the input and predicts its author. In this model, the input unit is a sentence. For each timestamp, a complete sentence represented by the average of all its word vectors is sent to the model. Same as the sentence-level model, the hidden state will be updated and an output will be produced upon each input. All of the outputs will be sent to an average pooling layer when no more input appears, whose output is the average of its input. Finally, a softmax classifier works on the output of the average pooling layer to make the decision which author wins out. The model structure is shown in FIGURE 2.4.

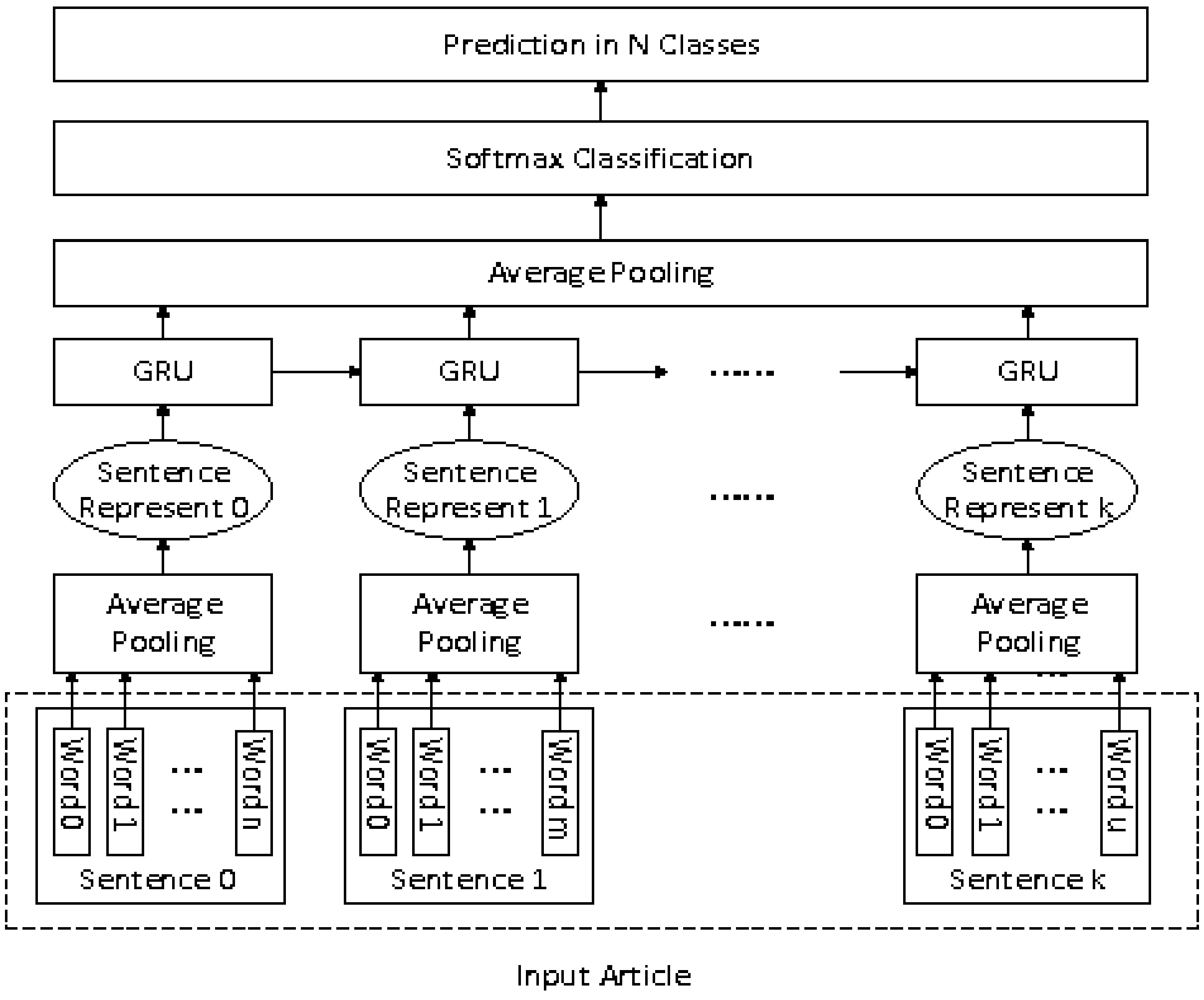


**Figure 2.4:** Article-Level Gru Diagram

### 2.1.6 Article-Level LSTM

Article-level LSTM is quite similar to the article-level GRU, sharing the same type of input and datan representation, and even the structure of the model, as shown in FIGURE 2.5. The only difference lies in the cell in the LSTM model.

### 2.1.7 Result and Limitation

They tested the Sentence Level GRU model using the C50 dataset. They varied the hidden size of the GRU network from 50 to 300. It shows that the larger the hidden size, the higher the accuracy. However, this model is problematic. The test accuracy is too low compared to training accuracy. The test accuracy is increasing as training accuracy increases. They concluded that this model has too little input and too many parameters and it could lead to overfitting on the training set.
Then they tested the model for Article-Level GRU, By comparing the results, they concluded that a hidden size of 150 and a regularization of 0.00005 might be a reasonable choice since they have the highest test accuracy. The best accuracy for C50

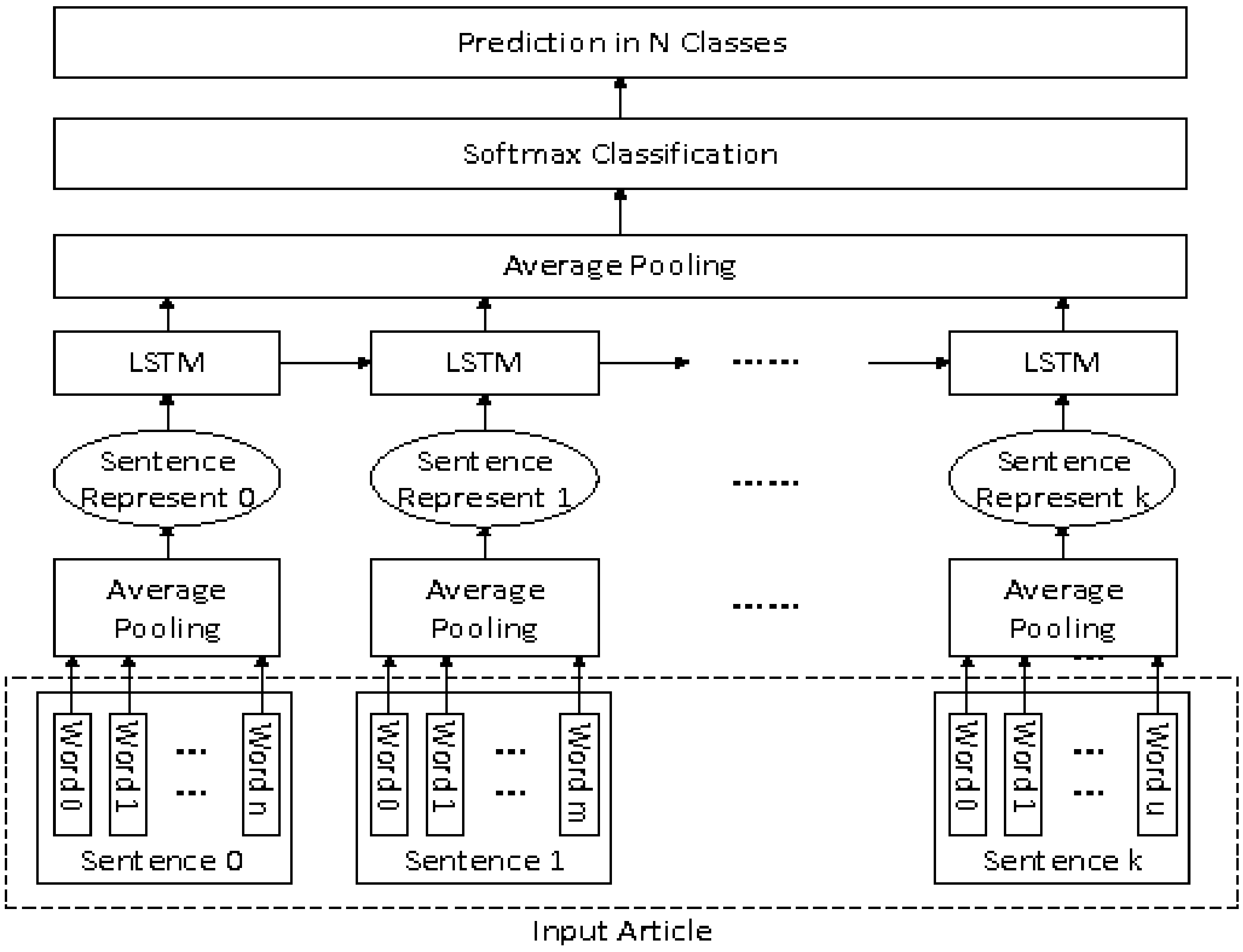


**Figure 2.5:** Article Level LSTM Diagram

are 69.1%. The F1 score is 0.66.
On testing for model Article-Level LSTM, For this model, they also varied the hidden size with C50 dataset and the result is shown fig. the highest test accuracy of LSTM, however, is 62.7%, lower than what the previous model can achieve.

Therefore, Article-level GRU performed best on authorship identification, outputing an accuracy of 69.1% on C50.
There are two major limitations in their models. First, in the article level model, the strategy of representing a sentence is averaging the word vector, which ignores the temporal relationship among words, and also the inside structure of a sentence.
Second, if one word does not exist in the look-up dictionary, they simply replace that with the magic word. For articles containing a lot of unseen words, the performance of the model loses guarantee.

# Chapter 3

# Dataset

Bollywood, the Hindi film industry churns out numerous songs and movies every year. According to the reports on soundchart.com on average, around 10 Bollywood movies are released monthly and each of them has around 7-8 songs. This amounts up to a big approximate sum of 850 songs that come into the market on an annual basis which makes Bollywood one of the most popular forms of music in India. This is a rich source of textual data if we look at the scripts of the movies and song lyrics. They can be used for the creation of data resources that are valuable for a variety of applications. These applications can be based on the audio or the textual aspect of the songs. Some research has also explored multi-modal approaches. Lyrics, in particular aid in tasks related to music information retrieval, genre classification, sentiment analysis, opinion mining, code-mixing, etc. This motivated the creation of an annotated dataset for Bollywood lyrics. A dataset is helpful in stating a research question, testing the hypotheses, and evaluating the outcome. It is always important to clean and pre-process the data before starting with any experiments. Also, some experiments on the data make available insights on its suitability. That is how the work presented here, on Bollywood lyrics, took its course. This chapter comprises the collection and cleaning of the lyrics that gave rise to the Bollywood lyrics dataset. The market of Bollywood music is massive with a customer base of 14 million people and with the advancement in technology, digitization of music is inevitable. Bollywood music and lyrics are being made available online with a greater speed and outreach with every passing year. Bollywood lyrics can be easily extracted from many websites. They are made available in different formats, i.e., transliterated in Roman script or the original form, in the Devanagari script. For simplicity of processing, we extracted them from source 1 where Hindi words were available in Devanagari script, i.e. consisting of utf-8 characters. The lyrics of the songs were extracted with their metadata including the movie or album they belong to, the singers, and the lyricist.

## 3.1 Features of Dataset

The data consisting of 14000 lyrics of more than 900 lyricists was extracted. The lyrics are extracted from [gii, 2002] using the webpage crawler approach using [Richardson, 2007]. On exploring we found that only a few lyricists have a good number of count of lyrics. So we take the top 11 lyricists based on the count. We capped at 11 because the 11th lyricist was Gulzar and we don't want to leave Gulzar as he is one of the famous lyricists. So, a dataset consisting of around 6000 lyrics in the Devanagari script of the top 11 Bollywood lyricists was created for this purpose. The distribution of lyricists and the number of songs are displayed in Figure 3.1.

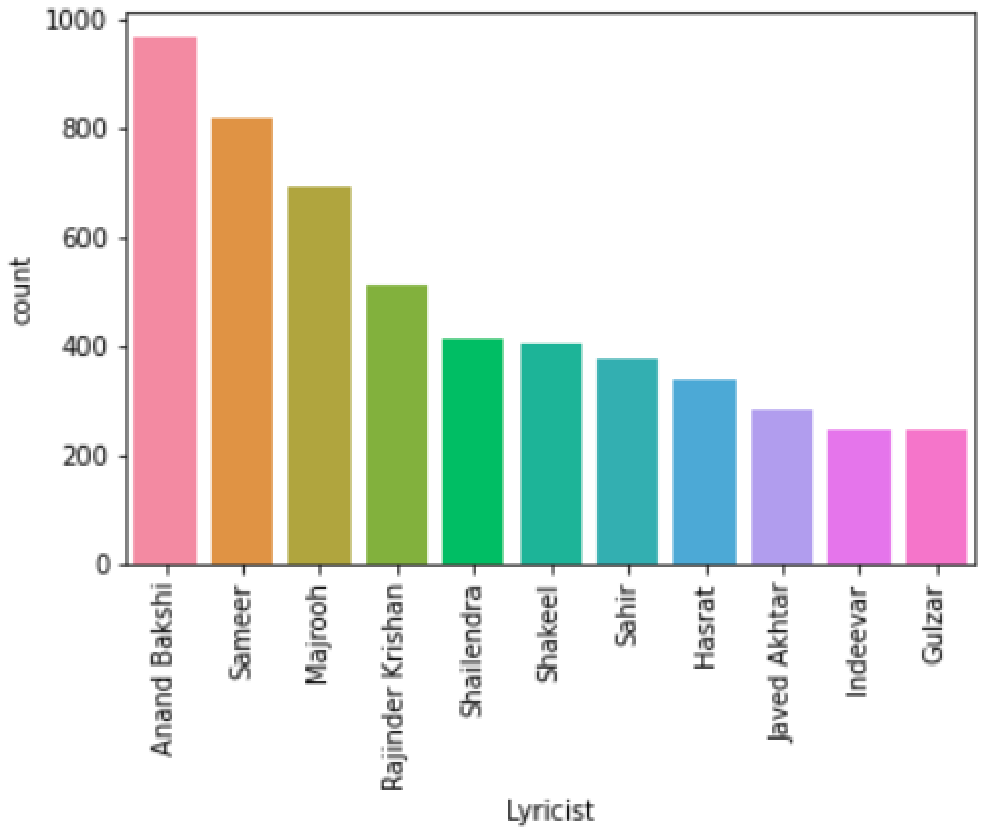


**Figure 3.1:** Lyricist Distribution

## 3.2 Dataset Preparation and Pre-Processing

In this part, we will discuss dataset preparation, We found that in our dataset in 14,000 lyrics we have around 100,000 words but we found that there are a lot of symbols, English alphabets and numeric were present in the vocab. We preprocessed the file to remove this noise and we also worked on removing stop words that are very common in Hindi words by creating a list of stop words ourselves. Thus after data pre-processing our vocab size has now decreased to about 70,000 words. This created the corpus for pretraining of our BERT. We created a tokenizer using Bert word piece tokenizer to tokenize the text and map word into numeric. We have chosen a Vocab size of 40000 with a max sentence length of 200 and use post padding to create inputs in a specific format. As we have capped the number of lyricists to 11 we have around 6000 lyrics for the classification task. The dataset is split into train and test set with the ratio of 80:20 with stratifying by classes. After this, we faced the class imbalance in both train and test datasets. So to resolve this issue we use a random oversampling approach to make the class balance in both the dataset.

# Chapter 4

# Approach

In this section, the approaches of authorship identification we took are elaborated. We selected two models by [Hochreiter and Schmidhuber, 1997] and [Chung et al., 2014] architecture mentioned in [Narayanan, 2020] and taken it as our baseline model and compared it with the current state-of-the-art method in [Devlin et al., 2018]

## 4.1 Methodology

BERT, which stands for Bidirectional Encoder Representations from Transformers. BERT is designed to pre-train deep bidirectional representations from an unlabeled text by learning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as classification, question answering, and language inference, without substantial task-specific architecture modifications. BERT is conceptually simple and empirically powerful.
BERT's key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modeling. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. The paper's results show that a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models.

## 4.2 Working of BERT:

BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms, an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary. As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore it is considered bidirectional, though it would be more accurate to say that it's nondirectional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word). BERT uses two training strategies:

### 4.2.1 Masked LM (MLM)

Before feeding word sequences into BERT, 15% of the words in each sequence are replaced with a [MASK] token. The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence.
In technical terms, the prediction of the output words requires adding a classification layer on top of the encoder output. Multiplying the output vectors by the embedding matrix, transforming them into the vocabulary dimension. Calculating the probability of each word in the vocabulary with softmax.
The BERT loss function takes into consideration only the prediction of the masked values and ignores the prediction of the non-masked words. As a consequence, the model converges slower than directional models, a characteristic that is offset by its increased context-awareness.

### 4.2.2 Next Sentence Prediction (NSP)

In the BERT training process, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. During training, 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document, while in the other 50% a random sentence from the corpus is chosen as the second sentence. The assumption is that the random sentence will be disconnected from the first sentence. To help the model distinguish between the two sentences in training, the input is processed in the following way before entering the model. A [CLS] token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence. A sentence embedding indicating Sentence A or Sentence B is added to each token. Sentence embeddings are similar in concept to token embeddings with a vocabulary of 2. A positional embedding is added to each token to indicate its position in the sequence.
When training the BERT model, Masked LM and Next Sentence Prediction are trained together, to minimize the combined loss function of the two strategies.

### 4.2.3 BERT (Fine-tuning)

Using BERT for a specific task is relatively straightforward:
BERT can be used for a wide variety of language tasks, while only adding a small

layer to the core model. Classification tasks such as sentiment analysis are done similarly to Next Sentence classification, by adding a classification layer on top of the Transformer output for the [CLS] token.

## 4.3 Architecture

In this section, we will discuss the architecture of our model, as shown in Figure 4.1. The bottom-most layer is the input layer which inputs the text data in form of the Devanagari script. The input in our model is lyrics of song in text format in Devanagari script.

In this section, we will discuss the architecture of our model, as shown in Figure 4.1. The bottom-most layer is the input layer which inputs the text data in form of the Devanagari script. The input in our model is lyrics of song in text format in Devanagari script.

The input is passed through pre-processing step which removes stopwords and noise. The text is mapped to a numeric using BERT word piece tokenizer. In Bert's word piece tokenizer word can be broken down into more than one sub-words.

After the tokenization process, words are then passed through Bert pretrained layer. Bert pretrained layer is trained using two task, masked language model and next sentence predictions. Bert pretrained layer take input in combination of three embeddings, token embedding, position embedding and sentence embedding and this embedding is learned using masked language model and next sentence predictions. The size of embedding we have selected is 128.

The output of pretrained layer is then given to bert finetuned layer. Finetuned layer is a layer which is basically implemented based on the task we are performing. Here in our case we are doing a classification task. So, a set feed-forward network with a pooling layer is used and output is passed through dense layer. The dense layer of size of 11 is used because we have 11 classes in classification task. In both models, the output of the dense layer is converted into probability scores using the softmax function and based on maximum probability scores we predict the lyricist of the lyrics.

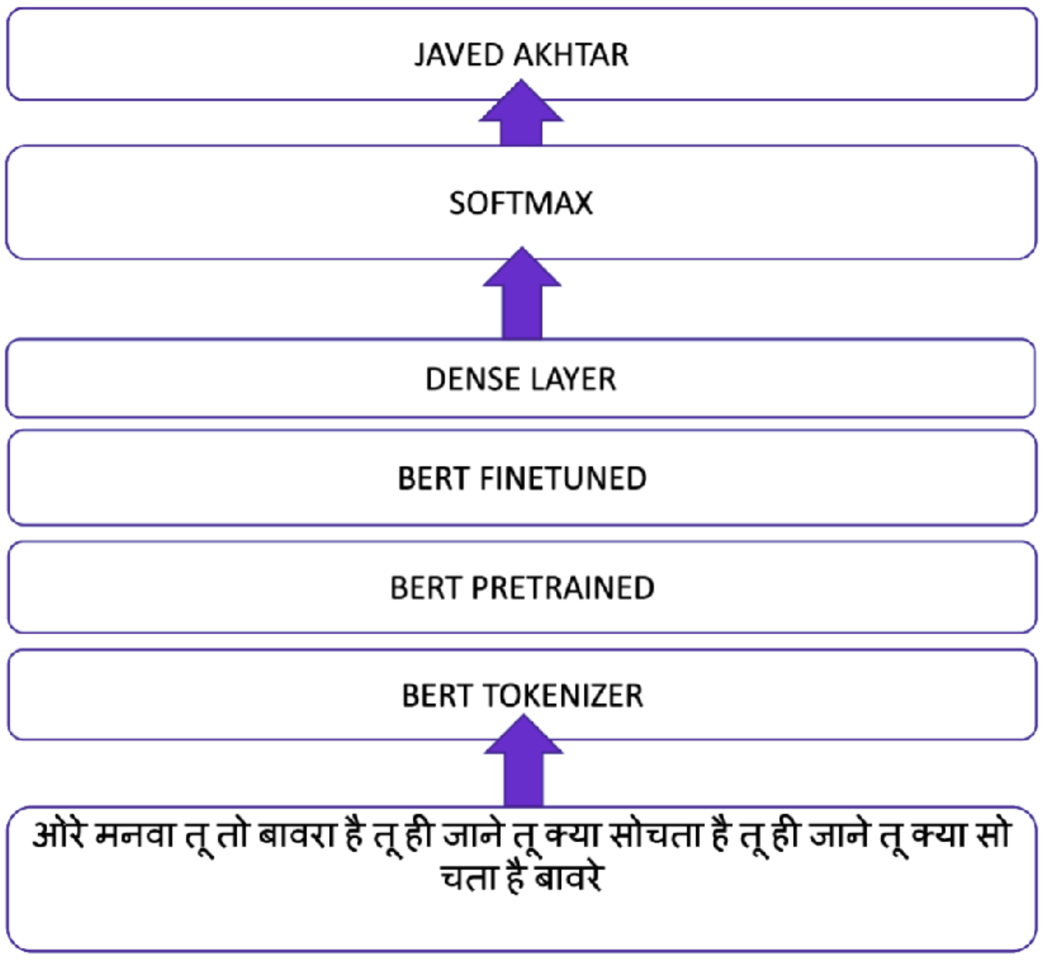

JAVED AKHTAR
SOFTMAX
DENSE LAYER
BERT FINETUNED
BERT PRETRAINED
BERT TOKENIZER
ओरे मनवा तू तो बावरा है तू ही जाने तू क्या सोचता है तू ही जाने तू क्या सोचता है बावरे

**Figure 4.1:** Pipeline showing the inputs and outputs of BERT classifier

# Chapter 5

# Experiments And Result

In this section, we elaborate on the deep learning models we used in our experiment. In total three models are implemented: two models are based on [Narayanan, 2020] i.e article-level LSTM and article-level GRU which will be the baseline model and we will compare the baseline accuracy with BERT.
Before experimenting, we split the dataset to create a training and test set for evaluation of our model. Further, to overcome the imbalanced classes we used random sampling on train and test.
To compare BERT with article-level LSTM and GRU, we chose a text classification task and trained both BERT and article-level LSTM and GRU on the same training sets, evaluated with the same test sets.
As can be seen in Figure 4.1, the classifiers take lyrics in text format and predict the lyricist. Each lyric has a lyricist label in the dataset and the goal of the classifiers is to predict the lyricist of lyrics.

For tuning the BERT model, a learning rate of $2xe^{-5}$ was used with adam optimizer. We selected the bert base model in our architecture. Bert base model contains 12 stacked encoders in its implementation. Also in training BERT we have not used any pre-trained model and built our model from scratch using domain-specific corpus based on Hindi songs in Devanagari script. In total, a vocab size of 40000 unique words with a maximum sentence length of 200 words and embedding size of 128 is selected. With this configuration, we trained the Bert model for 5 epochs. After the pretrained layer is trained, we used the pretrained layer for the classification task by applying two dense layers with an average pooling layer between them, and on top of this layer, a softmax is used to get the prediction probabilities. The finetuned layer is run for 50 epochs with adam optimizer. In the experiment, we found that BERT performed well with giving an accuracy of 83% with a confusion matrix of 11 lyricists, and precision and recall for each class of the BERT model is shown in Figure 5.1, and Figure 5.2.

For evaluation purposes we have used three metrics Accuracy, Precision, Recall where

Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$

Precision = $\frac{TP}{TP+FP}$

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

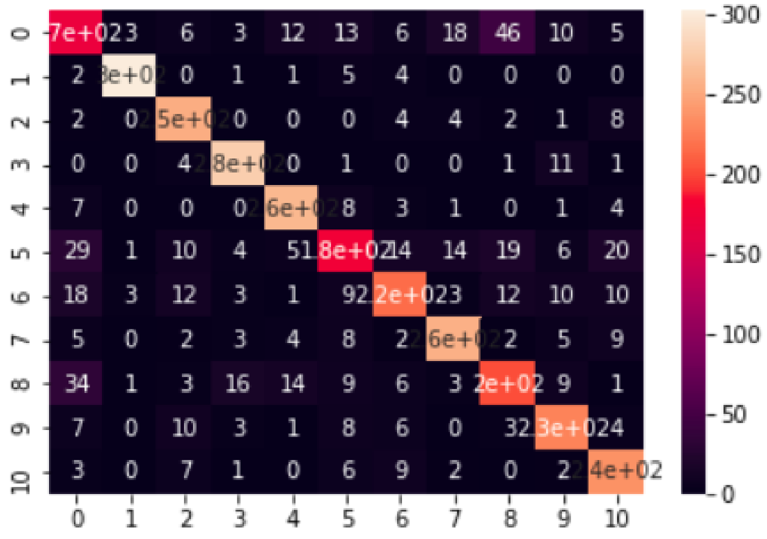
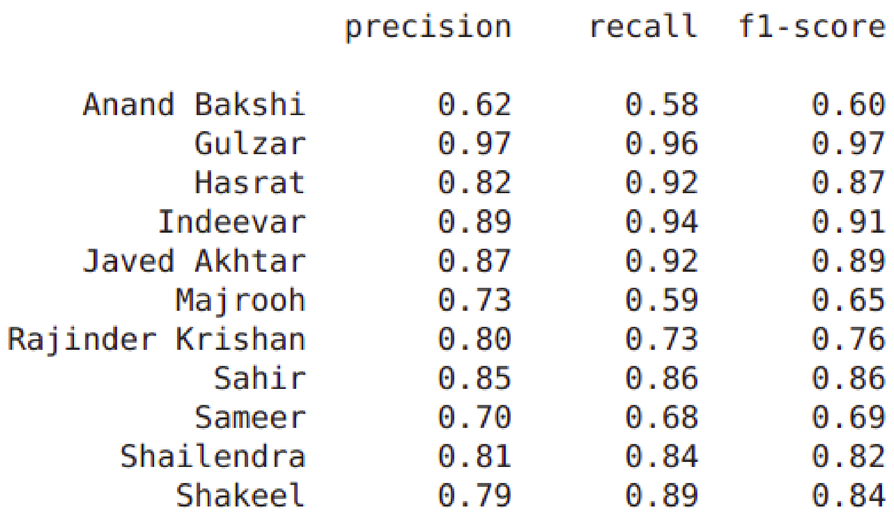


**Figure 5.1:** Confusion Matrix of BERT



```
                  precision    recall  f1-score

    Anand Bakshi       0.62      0.58      0.60
          Gulzar       0.97      0.96      0.97
          Hasrat       0.82      0.92      0.87
        Indeevar       0.89      0.94      0.91
    Javed Akhtar       0.87      0.92      0.89
         Majrooh       0.73      0.59      0.65
Rajinder Krishan       0.80      0.73      0.76
           Sahir       0.85      0.86      0.86
          Sameer       0.70      0.68      0.69
      Shailendra       0.81      0.84      0.82
          Shakeel       0.79      0.89      0.84
```

**Figure 5.2:** Precision and Recall of the BERT model

While training with the article level LSTM and GRU model, a learning rate of 0.01 was used with Adam optimizer with 50 epochs. The best accuracy of article-level GRU was 76%, and article-level LSTM was 68% as shown in Figure 5.3.
Below is the comparison of Accuracy of BERT, article-level LSTM, and GRU in Figure 5.5,

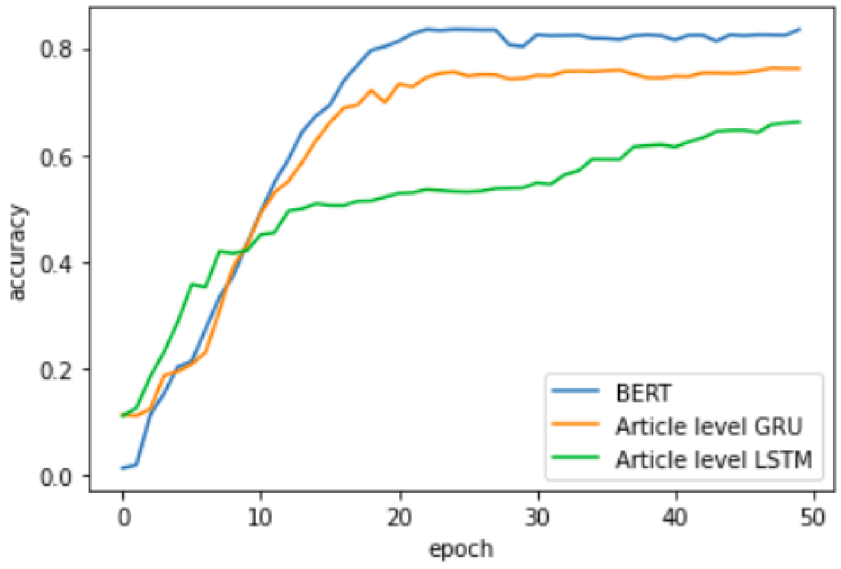


**Figure 5.3:** Bert, article-level LSTM, article-level GRU Accuracy

We also experimented with pre-trained embeddings, we used a word2vec embedding which is based on Skip-Gram embedding trained on Hindi CoNLL17 corpus, with a vocabulary of 0.2M words and size 100 dimensions, trained using a window size of 10 in the work of [Ginter et al., 2017].
But the results were reduced using these embedding, with BERT we got an accuracy of 76%, and with article-level LSTM, and GRU the accuracy was found to be 32% and 28% as shown in Figure 5.4 respectively. The main reason for this accuracy is found because of out of vocabulary words in word2vec embedding.

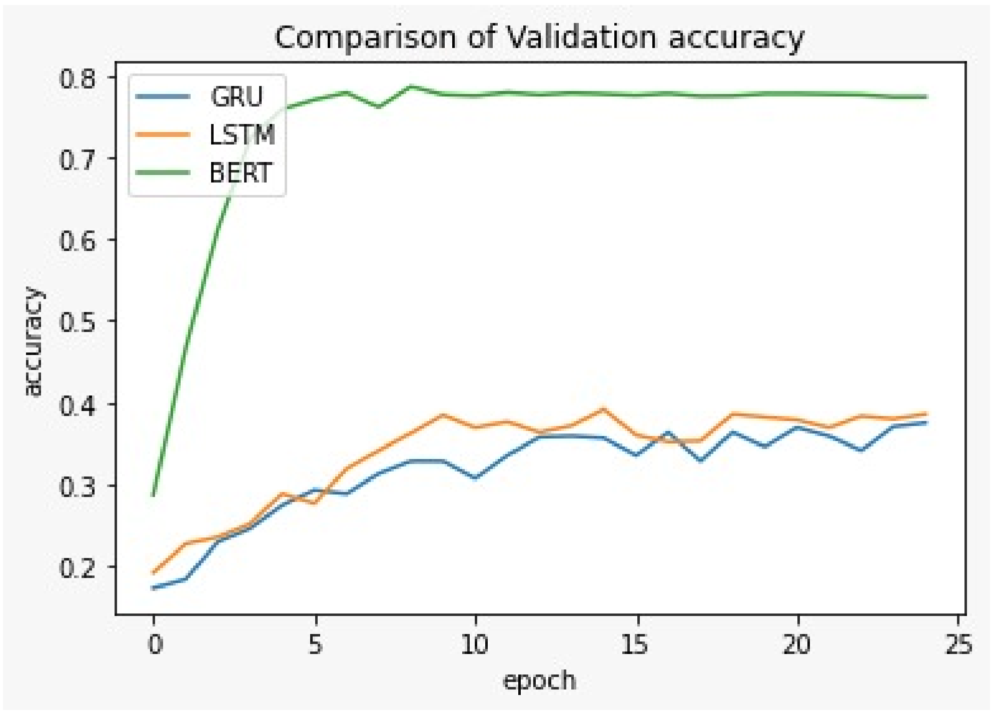
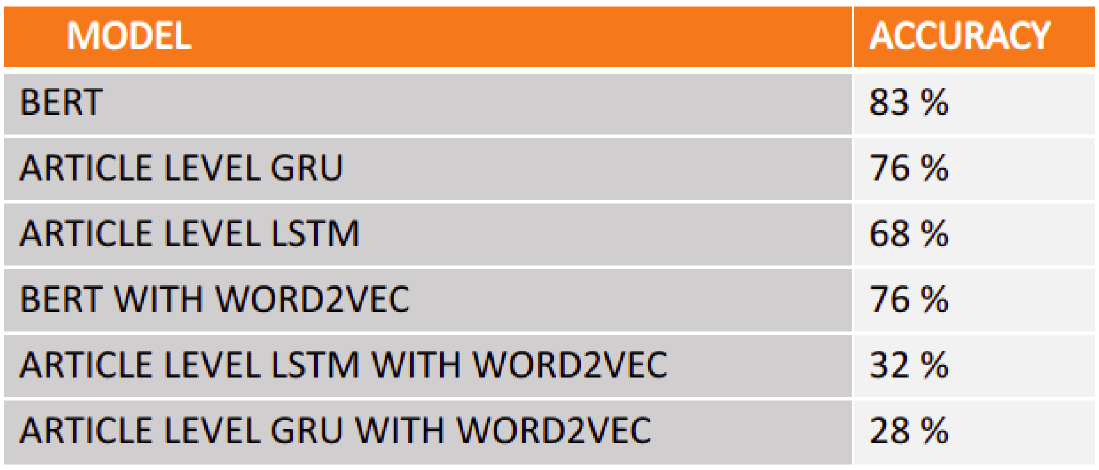


**Figure 5.4:** Bert, article-level LSTM, article-level GRU on word2vec embedding



| MODEL | ACCURACY |
|---|---|
| BERT | 83 % |
| ARTICLE LEVEL GRU | 76 % |
| ARTICLE LEVEL LSTM | 68 % |
| BERT WITH WORD2VEC | 76 % |
| ARTICLE LEVEL LSTM WITH WORD2VEC | 32 % |
| ARTICLE LEVEL GRU WITH WORD2VEC | 28 % |

**Figure 5.5:** COMPARING ACCURACY

# Chapter 6

# Conclusion

In this project, we studied different deep learning models on authorship identification on Bollywood lyrics data which were in Devanagari script.

- We designed and implemented 3 models for authorship identification, on our lyrics dataset. BERT performed best on authorship identification, giving us an accuracy of 83% on the Bollywood lyrics dataset. Whereas our baseline model article-level GRU and article-level LSTM presented by [Narayanan, 2020] gave an accuracy of 76% and 68% respectively.
- The reason for having low accuracy of article-level LSTM and GRU is because of their learning in only one direction and also it can be because BERT learns the context better than recurrent networks.
- We also tried transfer learning by training our model on [Mikolov et al., 2013] embedding but found that accuracy decreases for all the 3 models we have used.
- The reason behind this is lack of vocabulary i.e embedding for some words that were not in the vocabulary. But as we set the embedding layer as trainable the accuracy increased but still, it was less than our original model.

# Chapter 7

# Future Work

Many future works could be done, such as :

- Trying different sizes of word embeddings, exploiting size of vocab, trying different learning rates and epochs for pretraining of our BERT model.
- Using pre-trained BERT model for our classification task.
- In the Bert model, there is also a scope of research in pretraining on some big corpus or taking deep architecture on fine-tuning stage.
- There is also a great amount of meaningful work worth studying in areas like finding the relationship between two lyricists, or the sentiment analysis or gender bias in lyrics.

# References

[gii, 2002] (2002). Bollywood song lyrics. `https://www.giitaayan.com/`.

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling.

[Devlin et al., 2018] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

[Ginter et al., 2017] Ginter, F., Hajič, J., Luotolahti, J., Straka, M., and Zeman, D. (2017). CoNLL 2017 shared task - automatically annotated raw texts and word embeddings. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

[Narayanan, 2020] Narayanan, R. (2020). Deep learning based authorship identification.

[Richardson, 2007] Richardson, L. (2007). Beautiful soup documentation. *April*.