Shahjalal University of Science and Technology

Institute of Information and Communication Technology

Punctuation Restoration For Automatic Speech Recognition System Using Transformer Model

MD HABIBUR RAHMAN

Reg. No.: 2017831002

4th year, 2nd Semester

MEHEDI HASAN

Reg. No.: 2017831040

4th year, 2nd Semester

Department of Software Engineering

Supervisor

DR MOHAMMAD SHAHIDUR RAHMAN

Professor

Department of Computer Science and Engineering

21st August, 2023

Punctuation Restoration For Automatic Speech Recognition System Using Transformer Model

A Thesis submitted to the Institute of Information and Communication Technology, Shahjalal University of Science and Technology, in partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering.

By

MD Habibur Rahman
Reg. No.: 2017831002
4th year, 2nd Semester

Mehedi Hasan
Reg. No.: 2017831040
4th year, 2nd Semester

Department of Software Engineering

Supervisor

Dr Mohammad Shahidur Rahman

Professor

Department of Computer Science and Engineering

21st August, 2023

Recommendation Letter from Thesis Supervisor

The thesis entitled *Punctuation Restoration For Automatic Speech Recognition System Using Transformer Mode* submitted by the students

1. MD Habibur Rahman

2. Mehedi Hasan

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

Signature of the Supervisor:

Name of the Supervisor: Dr Mohammad Shahidur Rahman

Date: 21st August, 2023

Certificate of Acceptance of the Thesis

The thesis entitled *Punctuation Restoration For Automatic Speech Recognition System Using Transformer Mode* submitted by the students

1. MD Habibur Rahman
2. Mehedi Hasan

on 21st August, 2023, hereby, accepted as the partial fulfillment of the requirements for the award of their Bachelor Degrees.

Director of the IICT.	Chairman, Exam. Committee	Supervisor
Prof M. Jahirul Islam, PhD., PEng.	Prof M. Jahirul Islam, PhD., PEng.	Dr Mohammad Shahidur Rahman
Professor	Professor	Professor
Institute of Information and Communication Technology	Institute of Information and Communication Technology	Department of Computer Science and Engineering

Abstract

The paper addresses the challenge of restoring punctuation in ASR systems, which is crucial for enhancing the readability of transcribed text and facilitating natural language processing tasks. While this problem has been tackled for high-resource languages like English, it is still an underexplored area for low-resource languages like Bangla. In this paper, we propose using transformer models, a type of deep learning model, to address this problem in Bangla, which is only the second reported work for this language. We used pre-trained language models like BERT, ALBERT, and RoBERTa, which are known for their effectiveness in natural language processing tasks, and created a new dataset for public use. The experimental results on the Bangla dataset showed that the transformer-based approach achieved the current state-of-the-art performance, indicating that this method can be a promising solution for the punctuation restoration problem in low-resource languages like Bangla.. We also prepared a new dataset published publicly for the research community. Experimental result on the Bangla dataset has proved that the transformer-based approach has acquired current state-of-the-art result.

Acknowledgements

At the outset of our research, we wish to convey our utmost gratitude to Allah for empowering us with the strength and composure needed to complete our study.

We would like to acknowledge the faculty and staff in the department for their support, encouragement, and guidance. Their expertise, insights, and feedback have been instrumental in shaping my research and academic journey.

We also would like to thank our families and friends for their love, support, and encouragement throughout the entire process. Their unwavering belief in us has been a source of motivation and inspiration.

Finally, We would like to express our sincere gratitude to our thesis advisor, **Dr Mohammad Shahidur Rahman** sir, for her guidance, feedback, and unwavering support throughout the entire research process. Her expertise, encouragement, and dedication have been invaluable in shaping our research and strengthening our skills.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Punctuation predictionrefers to the task of automatically predicting the appropriate punctuation marks in a given text, such as commas, periods, question marks, and exclamation points. The task is typically performed in scenarios where the original text does not contain any punctuation, such as in transcriptions of spoken language, or in situations where the text has been translated from a language that uses different punctuation conventions. Punctuation prediction is a challenging task in natural language processing, as it requires understanding the semantic and syntactic structure of the text to accurately predict the appropriate punctuation marks. A variety of techniques have been proposed for punctuation prediction, including rule-based methods, statistical models, and machine-learning approaches.

1.1 Motivation

Punctuation is a crucial element in conveying meaning and clarity in written language, and its absence or incorrect use can lead to ambiguity, confusion, and misinterpretation of the text. With the increasing availability of spoken language data and the need for automatic punctuation prediction methods, there is a growing interest in developing accurate and effective punctuation prediction techniques.

Furthermore, the ability to accurately predict punctuation can benefit language learners and individuals with language processing difficulties, providing additional cues for understanding and interpreting written language.

Therefore, the development of reliable punctuation prediction methods is an important area of research in natural language processing and has significant practical applications in areas such as machine translation, speech recognition, and information retrieval. The purpose of this thesis is to explore and evaluate various techniques for punctuation prediction and to contribute to the development of effective and accurate punctuation prediction methods.

ASR has made significant advancements in recent years. In ASR benchmarks like **TIMIT** [2] and **LibriSpeech** [3], state-of-the-art (SOTA) models have achieved extraordinarily low word error rates (WER), recorded at 8% and 1.4% Word Error Rates (WER), respectively. The emergence of voice assistants (including Siri, Cortana, Bixby, Alexa, and Google Assistant) and their increased consumer adoption were made possible by ASR's enhanced performance. To increase readability and use the transcriptions in later NLP applications, several components (such as acoustic and language models), pre-and post-processing procedures, and punctuation restoration are all necessary to be addressed.

The task of generating punctuation in the text has received a lot of attention in previous works, which can be broadly divided into three categories. The first approach, taken by some studies [4],[5],[6],[7], treats the task as a machine translation challenge. In this approach, a non-punctuated sequence is fed into the model, which generates an output sequence with appropriate punctuation marks. The second approach, proposed by some researchers [8], [9], [10], [11], [12] treats the task as a sequence labelling problem, where the model assigns a probability-based punctuation mark to each word in the input sequence. The third approach, proposed by others [13], sees the task as a token-level classification problem, where the model predicts a punctuation tag for each token in the input sequence. These different approaches highlight the diverse ways in which researchers have approached the challenge of generating punctuation in the text.

We looked at various transformer-based models for this study. To further enhance the performance of the models, On top of the transformer network that had already been trained we added a BiLSTM layer.

1.2 Problem Definition

Given,

Input sequences: $X = (x_1, x_2, \ldots, x_n)$

Punctuation tags: $Y = (y_1, y_2, \ldots, y_n)$

Now, the concept of punctuation restoration is a token-level classification task that produces a sequence. Like:

$\hat{Y} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n)$ in [O, COMMA, PERIOD, QUESTION],

where O denotes the label is **None** and The input utterance"s length is indicated by **n**. The predicted mark is automatically added to the position following the current token during the testing phase.

1.3 Organization of the Report

- Chapter 2: Literature Review of Existing works regarding Punctuation Prediction
- Chapter 3: This Chapter Contains the details about the Data-set
- Chapter 4: The proposed methodology is discussed in this chapter.
- Chapter 5: The result is summarized here.
- Chapter 6: Here we try to give the total picture of the report.

The dataset and code are publicly available here;

https://github.com/habib-wahid/punctuation-restoration

Chapter 2

Literature Review

2.1 Related Work

Recently, punctuation restoration for an Automatic Speech Recognition system has gained popularity. ASR systems usually produce unpunctuated text. So it is difficult to read the text. Recent standard approaches for punctuation prediction are deep neural networks.

Previously different paradigms are used to predict Punctuation. The first paradigm involves using [14,15]prosodic features (e.g., pitch, rhythm, intonation) to predict where punctuation should be inserted. However, this approach can be noisy and error-prone. The second paradigm combines prosodic features with lexical information (i.e., information about the words being spoken) to improve punctuation prediction. This approach can be effective but may require a dataset with both types of features, which may not always be available. The third paradigm relies solely on lexical information from transcripts to predict where punctuation should be inserted. This approach has been used in earlier attempts at punctuation restoration, where n-gram language models were used to predict hidden events.

Some of the earlier methods used lexical and prosody features, which means they looked at things like the words used in the text and how they were spoken (e.g. pauses, pitch, and intensity) to determine where punctuation marks should go. Newer approaches have incorporated acoustic characteristics, such as pitch, intensity, and pause duration, in combination with Hidden Markov Models (HMMs), to enhance the accuracy of punctuation restoration[16, 17]. These approaches have been found to work better when combined with textual data.

Recurrent neural networks (RNNs) and transformers have been shown to be effective for this task, as they can capture the contextual information of the input text[1, 18, 19]. Convolutional neural networks (CNNs) have also been used, as they can extract local features from the input text. The text mentions several model structures and techniques that have been developed to improve the accuracy of punctuation restoration. For example, recurrent neural networks (RNNs) and transformers have been shown to be effective for this task, as they can capture the contextual information of the input text. Convolutional neural networks (CNNs) have also been used[20, 21], as they can extract local features from the input text.

[22–24] Pre-trained language models such as BERT have also been used for punctuation restoration. These models are trained on large amounts of text and can capture the semantic and syntactic properties of language. By treating punctuation restoration as a sequence tagging task, where the model predicts the correct punctuation mark for each word in the input text, pre-trained language models have been able to achieve state-of-the-art performance on this task.

[25, 26] Even more recently, transformer-based approaches with pre-trained word embeddings have achieved state-of-the-art performance. For example, some researchers have used pre-trained BERT to restore punctuation, and others have studied various transformer architectures for PR and used an augmentation strategy that makes the models more robust to errors that can occur during automatic speech recognition (ASR).

In addition to these model structures, multi-task learning has also been utilized to improve the performance of punctuation restoration. Multi-task learning involves training a model to perform multiple related tasks simultaneously, which can improve its performance on each individual task. For example, sentence boundary detection and capitalization recovery are two related tasks that can be used to improve the performance of punctuation restoration.

In this paper, we aim to improve performance in a sequence labeling task by experimenting with different transformer-based models and propose a novel augmentation scheme. We draw inspiration from a previous study on text classification tasks and adapt their augmentation scheme to the specific needs of speech transcription. Specifically, we exclude certain techniques (synonym replacement and random swap) that are not relevant to this type of task. By doing so, we hope to improve the accuracy and effectiveness of our models for speech transcription applications.

Chapter 3

Dataset

There is a scarcity of resources for the Bangla punctuation restoration task, with only one small dataset publicly available. To address this, we have created a brand new dataset based on the publicly available corpus of Bangla articles. Our dataset includes a training set and a development set, each containing **20,000** articles selected from the train split of the corpus, as well as a test set containing 195 articles from the test split. The training set is the largest, with 4.1 million words, while the development set has **179,000** words and the test set has **87,700** words. This dataset will be a valuable resource for researchers working on the Bangla punctuation restoration task.

Bangla Dataset

DATASET	OVERALL	PERIOD	COMMA	QUESTION	OTHER
Training	4139081	275204(6.64 %)	165997 (4.73 %)	14570 (0.33 %)	3683313 (88.98 %)
Dev	178571	12961(7.34 %)	7454(4.21 %)	543 (0.3 %)	15723 (88.16 %)
Test in News	87700	6303(7.14 %)	4242(4.68 %)	345(0.35 %)	78061 (87.84 %)
Test in Ref.	6981	896(14.6 %)	309(4.09 %)	280(2.49 %)	5356(78.82 %)
Test in ASR	6387	807(13.82 %)	283(3.94 %)	135(1.95 %)	5092(80.29 %)

Table 3.1: Dataset Distribution

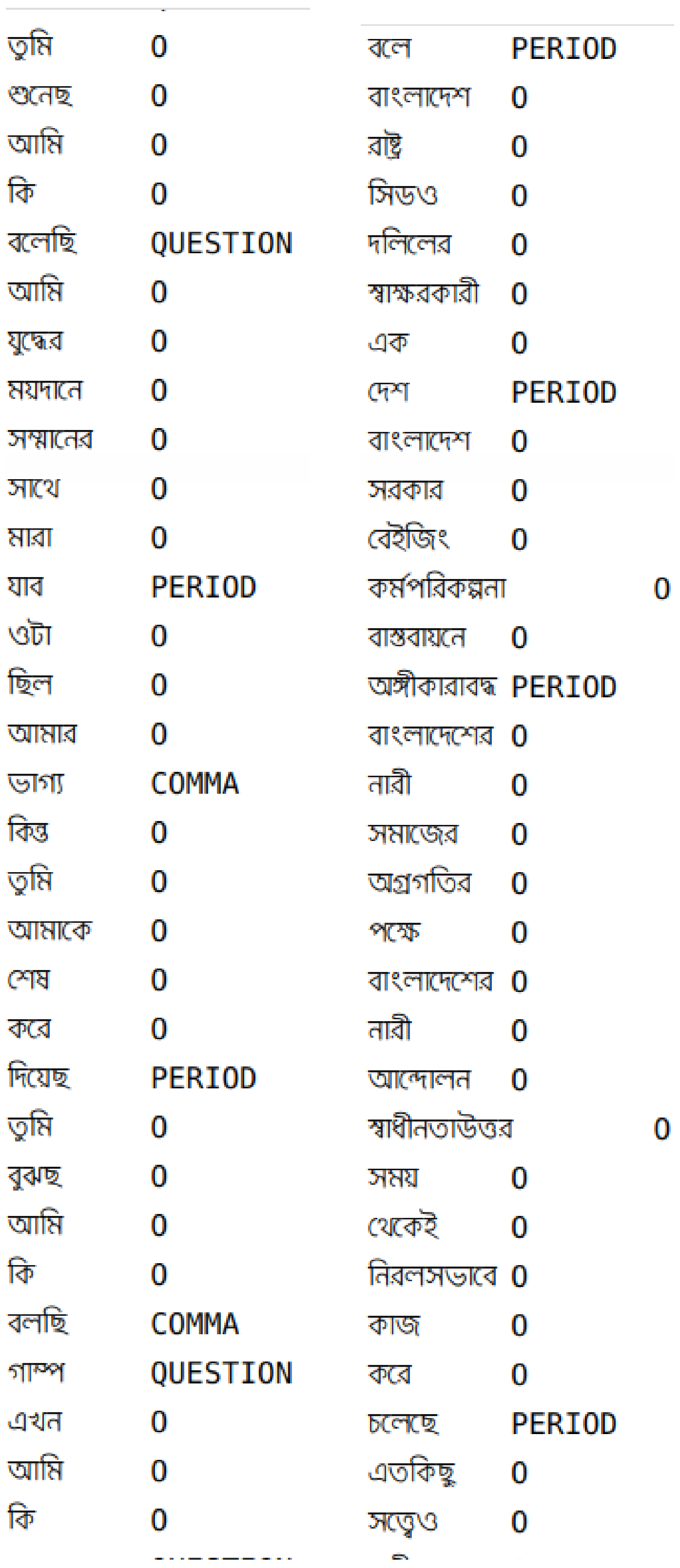
Two test datasets were gathered to evaluate the performance of the system. These datasets includes 65 minutes of speech collected from four Bangla short stories, which were read as monologues. The manual transcription of these excerpts, which included punctuation, was obtained. Additionally, ASR transcriptions were obtained using the Google Cloud speech API. However, the API did not provide punctuation for Bangla, so the ASR transcriptions were manually annotated with punctuation. To compute the Word Error Rate (WER), the ASR transcriptions were compared against the manual transcriptions, resulting in a 13.9% WER. The manual transcription consisted of 6793 words, while the ASR transcription consisted of 6510 words. Similar to English, the Bangla language has four punctuation marks: Period, Comma, Question, and O.

In our annotated dataset, "**,**" and "**:**" are replaced with "**COMMA**", "**।**", "**;**", "**!**" are replaced with "**PERIOD**", "**?**" is replaced with "**QUESTION**" and a word finishing with no punctuation is replaced with that word and "**O**".

Annotation Format

Regular word and Punctuation	Annotated Word and Punctuation
, :	COMMA
। ; !	PERIOD
?	QUESTION
Word	Word O

Table 3.2: Annotation Format Table



তুমি	O	বলে	PERIOD
শুনেছ	O	বাংলাদেশ	O
আমি	O	রাষ্ট্র	O
কি	O	সিডও	O
বলেছি	QUESTION	দলিলের	O
আমি	O	স্বাক্ষরকারী	O
যুদ্ধের	O	এক	O
ময়দানে	O	দেশ	PERIOD
সম্মানের	O	বাংলাদেশ	O
সাথে	O	সরকার	O
মারা	O	বেইজিং	O
যাব	PERIOD	কর্মপরিকল্পনা	O
ওটা	O	বাস্তবায়নে	O
ছিল	O	অঙ্গীকারাবদ্ধ	PERIOD
আমার	O	বাংলাদেশের	O
ভাগ্য	COMMA	নারী	O
কিন্তু	O	সমাজের	O
তুমি	O	অগ্রগতির	O
আমাকে	O	পক্ষে	O
শেষ	O	বাংলাদেশের	O
করে	O	নারী	O
দিয়েছ	PERIOD	আন্দোলন	O
তুমি	O	স্বাধীনতাউত্তর	O
বুঝছ	O	সময়	O
আমি	O	থেকেই	O
কি	O	নিরলসভাবে	O
বলছি	COMMA	কাজ	O
গাল্প	QUESTION	করে	O
এখন	O	চলেছে	PERIOD
আমি	O	এতকিছু	O
কি	O	সত্ত্বেও	O

Figure 3.1: Sample Bangla Dataset

Chapter 4

Methodology

In this study, we investigate multiple **transformer-based** models that support multilingual capabilities. Unlike previous research that focused solely on one architecture BERT [27], we explore various models. Additionally, we propose a novel augmentation technique that significantly enhances performance. Our augmentation method is similar to the techniques introduced in a previous paper, which involve synonym replacement, random insertion, random swap, and random deletion[28].

Before entering to main topic lets explore some thing about **Transformer** Architecture.

4.1 Transformer:

The Transformer is a deep learning model architecture that was introduced in the 2017 paper **Attention Is All You Need** [29]. It is primarily used in natural language processing (NLP) tasks such as language translation, language modelling, and text classification.

Before we explore the magnificent Transformer architecture, let's talk about some **historical context**.

4.1.1 Seq2seq

Deep learning models known as *Sequence-to-sequence* (abbreviated as **Seq2Seq**) [30] have achieved significant success in various tasks, such as machine translation, text summarization, and image captioning. Google Translate implemented a Seq2Seq model in their production system in late 2016. The fundamental principles of these models are described in two influential research papers. (Sutskever et al., 2014 [31], Cho et al., 2014 [32]).

The architecture of a **Seq2Seq** model consists of two main components, an **encoder** and a **decoder**. The encoder is responsible for understanding and capturing the contextual information of the input sequence in a hidden state vector. The decoder takes this vector as input and generates the corresponding output sequence. Due to the sequential nature of the task, both the encoder and decoder typically employ some form of **RNNs, LSTMs, GRUs,** or other similar recurrent neural network models.

4.1.1.1 Shortcomings of seq2seq:

The Seq2Seq model, despite its usefulness in natural language processing tasks, is not without its shortcomings.

- **Vanishing gradients**: One major limitation is that the Seq2Seq model can suffer from vanishing gradients, which can hinder its ability to learn effectively. Vanishing gradients occur when the gradients become extremely small during training, and the model struggles to update its parameters properly. As sequence size increases, model performance decreases.
- **Sensitivity to rare or out-of-vocabulary words :**Another drawback of Seq2Seq is its sensitivity to rare or out-of-vocabulary words. Since the model uses fixed-length vector representations, it may struggle to handle words that are infrequent in the training data, which can lead to the inaccurate or incomplete output.
- **Computationally expensive :** Finally, the Seq2Seq model can be computationally expensive to train, requiring large amounts of data and computational resources. This can limit its practical use in certain scenarios where resources are limited.

4.1.2 Seq2Seq with Attention:

To overcome the drawbacks of the **Seq2Seq** model a solution was proposed by Bahdanua,Loung in this two papers [33] [34]. These two publications introduced and refined the concept of **Attention** This technique allowed for a considerable improvement in machine translation systems by focusing on the relevant parts of the input sequence.

- Attention is a layer that lets a model focus on whats important
- **Queries, Values, and Keys** are used for information retrieval inside the Attention layer
- Works for languages with very different grammatical structures

The **improvement** are stated here with this graph:

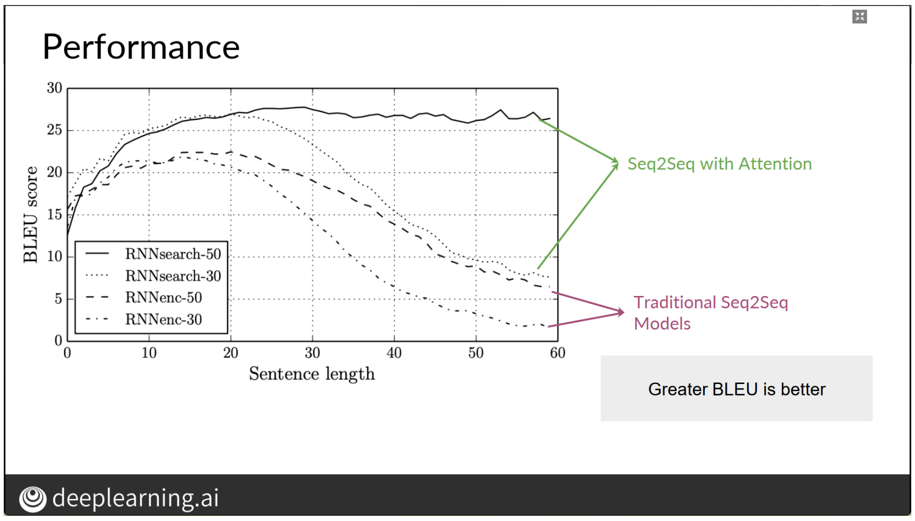


Figure 4.1: Improvement of Seq2Seq using Attention

4.1.3 Challenges and how Transformer models help to overcome:

- **Short-term Memory:** RNNs have a tendency to forget information from earlier time steps as they process new inputs. This makes it difficult for them to capture long-term dependencies in sequential data.

  Transformer Solution: Transformer networks almost exclusively use attention blocks. Attention helps to draw connections between any parts of the sequence, so long-range dependencies are not a problem anymore. With transformers, long-range dependencies have the same likelihood of being taken into account as any other short-range dependencies.

- **Sequential Computation:** RNNs are typically processed sequentially, which makes them slower than other neural network architectures like Convolutional Neural Networks (CNNs) and can be difficult to parallelize.

  Transformer Solution: Because transformers process their inputs in parallel, they can be faster and more efficient than RNNs. This is especially important for tasks that require processing large amounts of sequential data, such as machine translation.

- **Vanishing Gradient:** A well-known issue that occurs during the training of neural networks is the vanishing gradient problem.That happens when, as the loss function propagates backward through the layers of the network, the gradients of the loss function with respect to the network weights become incredibly small. As a result, during training, the weights of the network's lower layers are updated very slowly or not at all.

  Transformer Solution: Transformers use self-attention mechanisms to compute the representation of each token in the input sequence based on all the other tokens in the same sequence. This means that the gradients can be propagated through the entire sequence in one step, rather than through a chain of intermediate computations.

  Additionally, transformers use **skip connections** and **layer normalization**, which helps to stabilize the training process and reduce the impact of vanishing gradients. Skip connections allow the gradients to bypass the potentially problematic activation functions, and layer normalization helps to normalize the activations within a layer, making the network more resilient to the effect of vanishing gradients.

  Finally, the use of **residual connections** allows the gradient to flow directly to the lower-level layers without going through the higher-level layers, enabling the training of deeper models.

4.1.4 Attention Mechanism:

There are several different types of attention mechanisms used in transformers. Here are some of the most common:

4.1.4.1 Self-Attention:

The self-attention mechanism is a component of a model where tokens in a sentence interact with each other by looking at other tokens with an attention mechanism. This allows them to gather contextual information and update their own representations.

The operation can be described as a sequence-to-sequence process, where a sequence of vectors is inputted and a sequence of vectors is outputted. The input vectors are denoted as $x_1, x_2, \ldots, x_t$, and the corresponding output vectors are denoted as $y_1, y_2, \ldots, y_t$, all of which have a dimension of k. To compute the output vector y_i, the self-attention operation calculates a weighted average over all the input vectors, with the simplest option being the dot product.

In **self-attention**, each element of the input sequence is associated with three vectors: a **query** vector, a **key** vector, and a **value** vector. These vectors are learned during the training process and are used to compute the **attention** scores for each element in the sequence.

- **Query:** The query vector is used to determine which other elements in the sequence are most relevant to the current element. It is compared to the key vectors of all other elements in the sequence using a dot product, and the resulting scores are used to weight the contribution of each value vector to the final output.

- **Key:** The key vector is used to represent the element's position in the sequence and is used to compute the attention scores for other elements. The key vector is also compared to the query vector of the current element using a dot product to compute the attention scores.

- **Value:** The value vector is used to represent the content of the element and is used to compute the final output based on the attention scores. The value vectors of all elements in the sequence are combined using the attention scores to compute the final output for the current element.

$$Attention(\underset{\text{from}}{q}, \underset{\text{to}}{k, v}) = \overbrace{softmax\left(\frac{qk^T}{\underset{\text{vector dimensionality of K, V}}{\sqrt{d_k}}}\right)}^{\text{Attention weights}} v$$

Figure 4.2: Calculation of Attention Scores

4.1.4.2 Multi-head Attention

Typically, in order to understand a word's role in a sentence, it is necessary to understand how it relates to other parts of the sentence. This is crucial for both analyzing the source sentence and creating the target. The reason for Multi-Head Attention is that we must allow the model to concentrate on several tasks. Multi-head attention has multiple **heads** that each function independently rather than just one attention process. The **attention scores** in the preceding explanation is focused on the entire sentence at once. This would result in the same outcomes even if two sentences had the same words in a different order. Instead, we would like to attend to different segments of the words. We can give the self-attention greater power of discrimination, by combining several self-attention heads, dividing the words vectors into a fixed number (h, number of heads) of chunks, and then self-attention is applied on the corresponding chunks, using **Q, K and V** sub-matrices. This produces **h** different output **matrices of scores**.

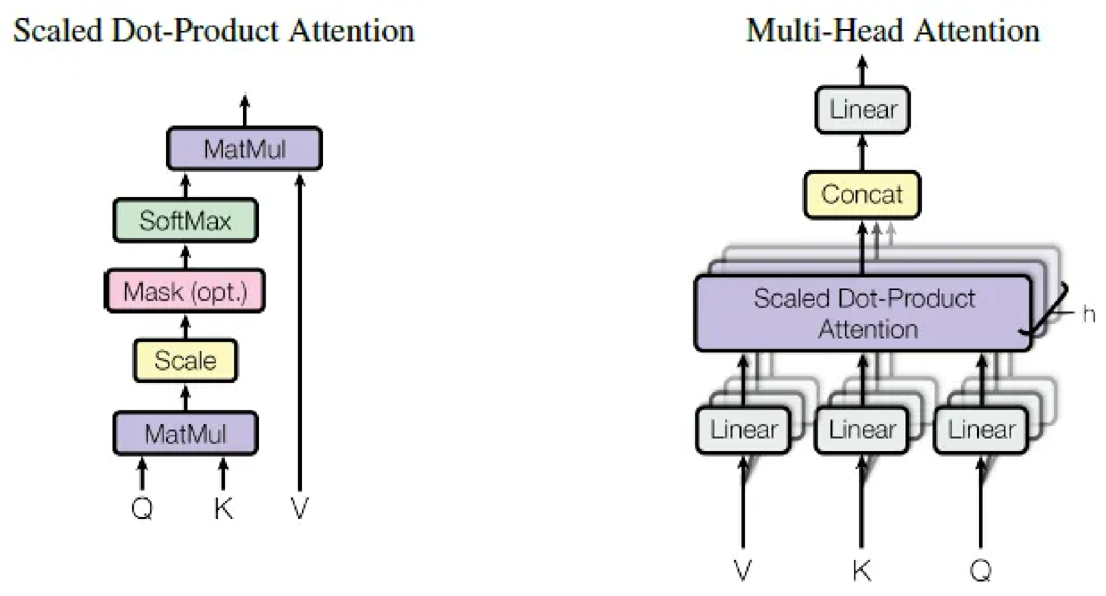


Figure 4.3: Multi-head Attention Mechanism

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_n)W_o$$

$$\text{head}_i = \text{Attention}(QW_Q^i, KW_K^i, VW_V^i)$$

4.1.5 Encoder:

The Transformer model's encoder produces an attention-based representation that enables it to identify specific information within an extremely vast context. Its structure is mainly composed:

- It is composed of a stack of N=6 identical layers
- Each layer consisting of **a multi-head self-attention** layer and a straightforward position-wise fully connected **feed-forward** network'
- Each layer of the Transformer model's encoder implements a **residual connection** and **layer normalization**. Moreover, the output of all sub-layers has the same dimension of 512.

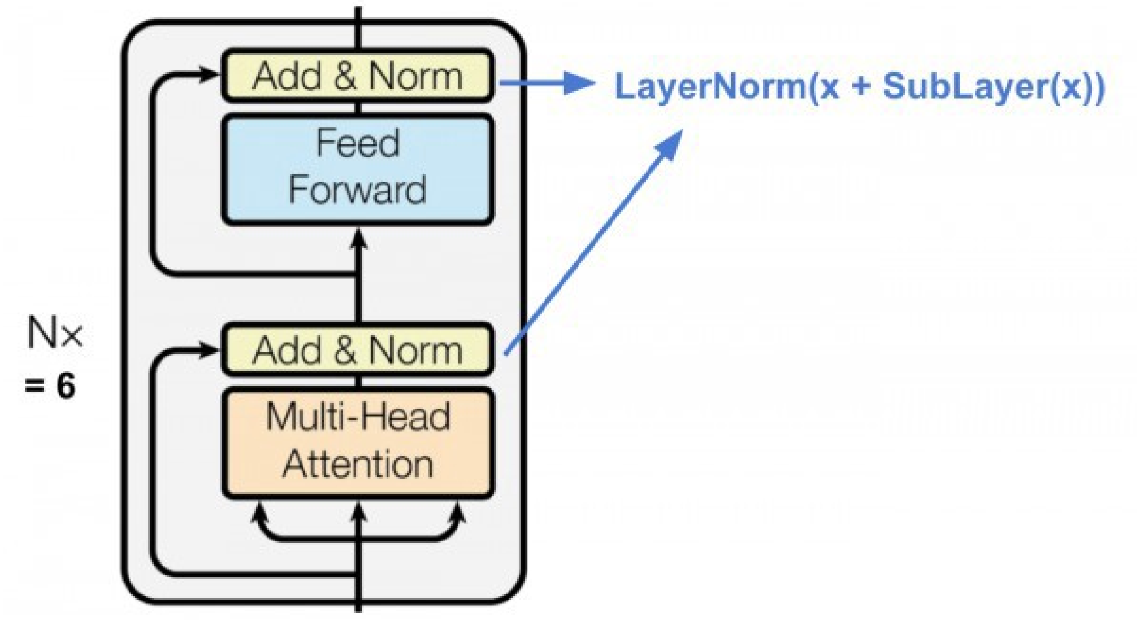


Figure 4.4: Encoder in Transformer [1]

In deep learning, a **residual connection** can be represented mathematically as $y = f(x) + x$, where $f(x)$ is the output of a sub-network and x is the input to that sub-network. The addition of x to $f(x)$ creates a shortcut that allows the network to bypass the sub-network if needed. Layer normalization, on the other hand, normalizes the inputs to each layer of the network by subtracting the mean and dividing by the standard deviation of the inputs. This can be expressed mathematically as $y = \frac{x-\mu}{\sigma}$, where x is the input, μ is the mean of the input, σ is the standard deviation of the input, and y is the normalized output. This normalization helps to stabilize the training process and improve the performance of the network. The above can be expressed in an equation:

$$y = LayerNorm(x + Sublayer(x))$$

Here The feed-forward network of the multi-head attention network is the sublayer.
Actually, the encoder block just performs a lot of matrix multiplications, followed by element-wise processing. Due to the fact that everything is simply **parallelizable** matrix multiplication, this procedure makes the transformer extremely quick. Hence, we build a very potent network by layering these modifications on top of one another.

4.1.6 Decoder:

The decoder is another important component in the transformer architecture used in natural language processing (NLP). The decoder is responsible for **generating the output** sequence based on the encoded input sequence. In other words, it takes the fixed-size representation produced by the encoder and generates a sequence of output vectors that correspond to the words or subwords of the output text.

Similar to the encoder, the decoder in the transformer architecture consists of a stack of self-attention and feedforward layers. However, unlike the encoder, the decoder also uses an additional type of attention mechanism called encoder-decoder attention. This allows the decoder to attend to different parts of the input sequence when generating the output sequence.

During training, the decoder is typically trained to predict the next word or subword in the output sequence based on the previously generated words. This is done using a technique called teacher forcing, where the decoder is fed the correct previous words during training. During inference, the decoder generates the output sequence autoregressively, using the previously generated words as input.

4.1.7 The Full Transformer Architecture:

- **Input Embedding:** The first step in the Transformer architecture is to convert the input sequence into a fixed-dimensional vector representation. This is done using an embedding layer that maps each token in the sequence to a continuous vector space.

- **Positional Encoding:** Since the Transformer architecture does not use recurrent or convolutional layers, it needs a way to capture the positional information of the input sequence. This is done using positional encodings, which are added to the embedded input sequence. The positional encodings provide information about the position of each token in the sequence.

- **Multi-Head Self-Attention:** The core of the Transformer architecture is the multi-head self-attention mechanism, which allows the model to attend to different parts of the input sequence while computing the representation of each token. The self-attention mechanism computes a weighted sum of the input sequence, where the weights are computed based on the similarity between each token and all other tokens in the sequence.

- **Feedforward Network:** After the self-attention layer, a feedforward network is applied to each position in the sequence independently. The feedforward network consists of two linear layers with a non-linear activation function in between.

- **Layer Normalization:** To stabilize the learning process and improve generalization, layer normalization is applied after each sub-layer of the Transformer architecture. Layer normalization normalizes the activations of the sub-layers and helps to prevent the vanishing or exploding gradients problem.

- **Output Layer:** The final step in the Transformer architecture is to compute the output distribution over the target sequence. This is done using a linear layer followed by a **softmax** activation function. The output layer is trained to predict the correct target sequence given the input sequence.

In summary, the Transformer architecture consists of an input embedding layer, a positional encoding layer, a multi-head self-attention layer, a feedforward network, layer normalization, and an output layer. This architecture has been shown to achieve state-of-the-art performance on various natural language processing tasks.

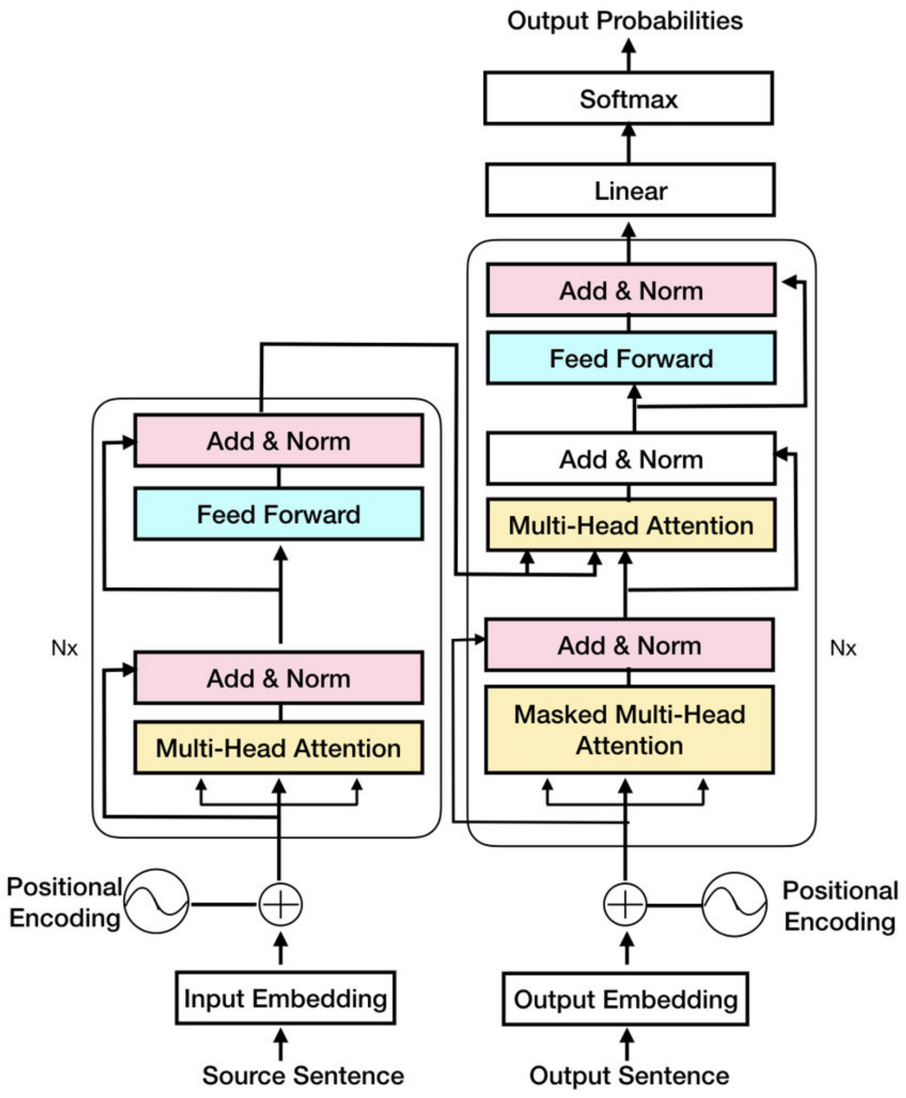


Figure 4.5: Transformer Architecture

4.2 Our Model Architecture

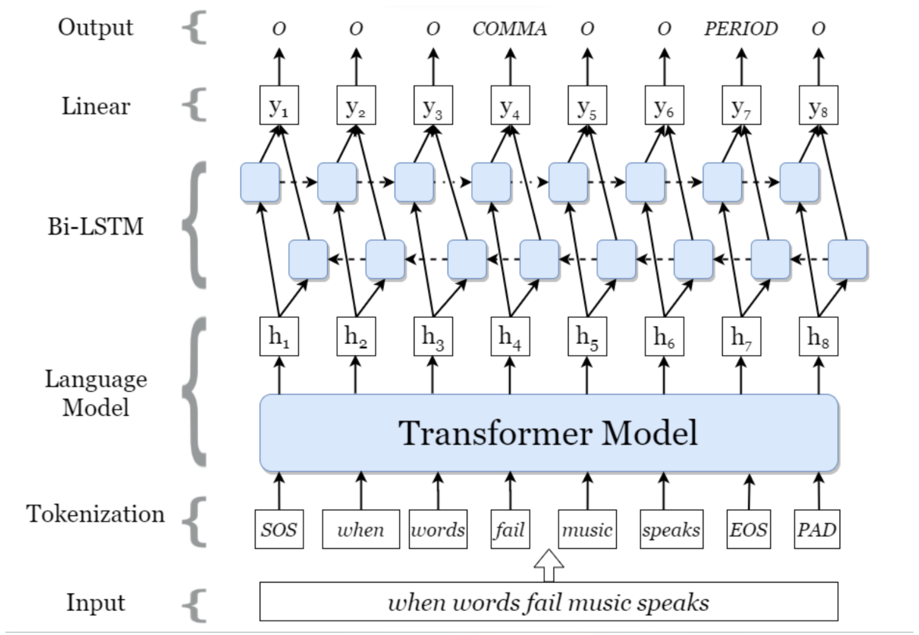

Output
O
O
O
COMMA
O
O
PERIOD
O
Linear
y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8
Bi-LSTM
h_1 h_2 h_3 h_4 h_5 h_6 h_7 h_8
Language Model
Transformer Model
Tokenization
SOS
when
words
fail
music
speaks
EOS
PAD
Input
when words fail music speaks

Figure 4.6: Our Exlored Model Architecture

Avobe figure illustrates the general architecture of our network used in the experiments. The pre-trained language model produces a d-dimensional embedding vector for each token, which serves as input for the BiLSTM [35] layer with h hidden units. This allows the network to consider both past and future contexts for prediction. The outputs from the forward and backward LSTM layers are concatenated and passed through a fully connected layer with four output neurons representing 3 punctuation marks and one O token. The input sentence does not have any punctuation, and the model's task is to predict the punctuation.

The models' performance is evaluated based on precision (P), recall (R), and F1-score (F1).

4.2.1 Augmentation

For this study, we propose an augmentation method inspired by the study of Wei and Zou (2019a), as discussed earlier. Our augmentation method is based on the types of error ASR makes during recognition, which include insertion, substitution, and deletion. Due to the lack of large-scale manual transcriptions, punctuation restoration models are typically trained using written text, which is well-formatted and correctly punctuated. Hence, the trained model lacks the knowledge of the typical errors that ASR makes. To train the model with such characteristics, we use an augmentation technique that simulates such errors and dynamically creates a new sequence on the fly in a batch. Dynamic augmentation is different from the traditional augmentation approach widely used in NLP [28] however, it is widely used in computer vision for image classification tasks [36]

The three different kinds of augmentation corresponding to three possible errors are as follows.

1. First (i.e., substitution), we replace a token by another token. In our experiment, we randomly replace a token with the special unknown token.

2. Second (i.e., deletion), we delete some tokens randomly from the processed input sequence.

3. Finally, we add (i.e., insertion) the unknown token at some random position of the input. We hypothesize that not all three errors are equally prevalent, hence, different augmentation will have a different effect on performance. When applying substitution, we replaced the token in that position with the unknown token and left the target punctuation mark unchanged. For deletion, both the token and the punctuation mark in that position are deleted. For insertion, we inserted the unknown token and O token, in that position. Since deletion and insertion operation may make the sequence smaller or longer than the fixed sequence length we used during training, we added padding or truncated as necessary.

Chapter 5

Experiment and Result

5.1 Result Discussion

In this section, we will discuss the evaluation of different multilingual models on a Bangla test set, which consists of news data, manual scripts, and ASR transcriptions. Due to the lack of a monolingual transformer model for Bangla, we explored the use of multilingual models and found that the XLM-RoBERTa (large) model performed the best. This model is specifically designed to be effective for low-resource languages like Bangla, as it is trained on a larger corpus of text and has a larger vocabulary for such languages.

In addition to using the XLM-RoBERTa model, we also applied data augmentation to improve performance on the Bangla test set. Specifically, we used augmentation parameters $\alpha = 0.16, \alpha sub = 0.2, and, \alpha del = 0.6$.

However, we noted that the performance of the XLM-RoBERTa model without augmentation is already strong, indicating that the model is well-suited for the Bangla language.

We also observed that the performance of the XLM-RoBERTa model was better on the news test set compared to the manual and ASR data. Furthermore, we found that the model's performance on Commas was lower than on Period and Question and that there was a performance drop of around 10% for Period and Question compared to the state-of-the-art result in the English dataset. However, for Comma, the performance drop was even more significant at over 30% on the ASR test set. These findings suggest that the XLM-RoBERTa model may be better suited for certain types of text and punctuation in the Bangla language and that further research is needed to better understand the model's performance on different types of data.

5.2 Result Table

Test	Model	Period			Comma			Question			Overall		
		P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
News	BERT-base-multilingual-uncased	79.8	68.2	73.5	80.4	85.4	82.8	72.1	77.0	74.5	79.9	78.5	79.2
	DistilBERT-base-multilingual-cased	72.1	60.8	66.0	74.5	71.6	73.0	56.9	67.5	61.8	73.0	67.3	70.1
	XLM-MLM-100-1280	76.9	71.2	73.9	82.0	83.4	82.9	70.2	76.4	73.2	80.0	78.5	79.3
	XLM-RoBERTa-large	86.0	77.0	81.2	89.4	92.3	90.8	77.4	85.6	81.3	87.8	86.2	87.0
	XLM-RoBERTa + augmentation	85.8	77.5	81.4	88.8	92.5	90.6	77.9	86.6	82.0	87.4	86.6	87.0
Ref.	BERT-base-multilingual-uncased	35.6	34.4	35.0	67.4	64.7	66.0	39.8	28.8	33.4	58.5	54.6	56.5
	DistilBERT-base-multilingual-cased	32.6	31.5	32.1	64.0	50.2	56.3	32.5	14.7	20.2	54.3	42.4	47.6
	XLM-MLM-100-1280	33.4	39.8	36.3	70.3	64.0	67.0	42.4	22.9	29.8	59.2	54.5	56.7
	XLM-RoBERTa-large	39.3	36.9	38.1	76.9	81.4	79.1	54.3	58.8	56.5	67.6	70.2	68.8
	XLM-RoBERTa + augmentation	43.3	37.3	40.1	76.5	82.6	79.4	53.0	56.5	54.7	68.3	70.8	69.5
ASR	BERT-base-multilingual-uncased	29.3	30.0	29.7	60.6	60.2	60.4	36.1	38.4	37.2	51.7	52.0	51.9
	DistilBERT-base-multilingual-cased	29.0	33.6	31.1	62.6	50.6	56.0	31.3	20.8	25.0	51.2	44.3	47.5
	XLM-MLM-100-1280	31.2	38.7	34.6	63.4	59.5	61.4	32.0	24.8	27.9	52.8	51.9	52.4
	XLM-RoBERTa-large	38.3	35.6	36.9	69.2	77.2	73.0	38.5	52.0	44.2	60.3	66.4	63.2
	XLM-RoBERTa + augmentation	37.2	33.2	35.1	69.1	77.8	73.2	45.5	60.8	52.1	61.1	67.2	64.0

Table 5.1: Result Table

5.3 Class Merging

We did a study that aims to improve the process of manual annotation in applications such as semi-automated subtitles generation by identifying the correct position of punctuations in the Bangla language. The study uses a model based on the XLM-RoBERTa (large) architecture coupled with augmentation techniques to predict the position of punctuations. The evaluation of the model is carried out on Three-Classes and Two-Classes test sets.

The Three-Classes test sets combine Period and Question to form one class, and Comma is included to form a second class. The third class represents no punctuation. The Two-Classes test sets combine Commas with the no punctuation class to form a single class of punctuated and unpunctuated sentences.

The results of the study show that the model performs well in predicting punctuation positions. The study found that there was a significant improvement in the performance of the model for manual and ASR transcriptions when reducing the number of classes from four to two. This improvement is attributed to the reduction in complexity of the classifier as the number of classes decreases. The performance gain decreased when merging four classes into three classes, but it boosts up significantly when reduced to two.

The findings of the study suggest that this type of model can help human annotators in applications such as semi-automated subtitles generation by reducing their time and effort in manually annotating punctuations.

Class Merging Table

Dataset	**Four-Classes**	**Three-Classes**	**Two-Classes**
	P R F_1	P R F_1	P R F_1
NewsText	89.1 85.3 88.0	87.0 84.2 89.6	94.0 92.9 94.0
Reference	67.9 71.9 70.5	73.0 75.8 76.6	76.2 84.6 87.0 86.1
ASR Text	60.1 68.2 65.0	65.1 72.5 67.1	76.0 84.7 81.0

Table 5.2: Merging classes result on Bangla Dataset

5.4 Ablation Studies

In natural language processing, it is common to use a linear layer followed by a softmax layer for sequence tagging tasks such as named entity recognition or part-of-speech tagging. This approach allows for the prediction of the most probable tag for each token in a sequence, based on the output probabilities of the softmax layer.

However, some researchers have also explored using Conditional Random Fields (CRF) for sequence tagging. CRF is a probabilistic model that takes into account the dependencies between adjacent tags in a sequence, which can lead to more accurate predictions. In this approach, the output of the linear layer is used as input to the CRF layer, which predicts the most probable tag sequence.

In the passage you provided, we experimented with using CRF in place of the softmax layer, but found that it did not improve performance on the task they were working on (which was not specified in the passage). In fact, we observed a slight decrease in performance. This could be due to a variety of factors, such as overfitting or the CRF layer not being able to effectively capture the dependencies between tags in the sequence.

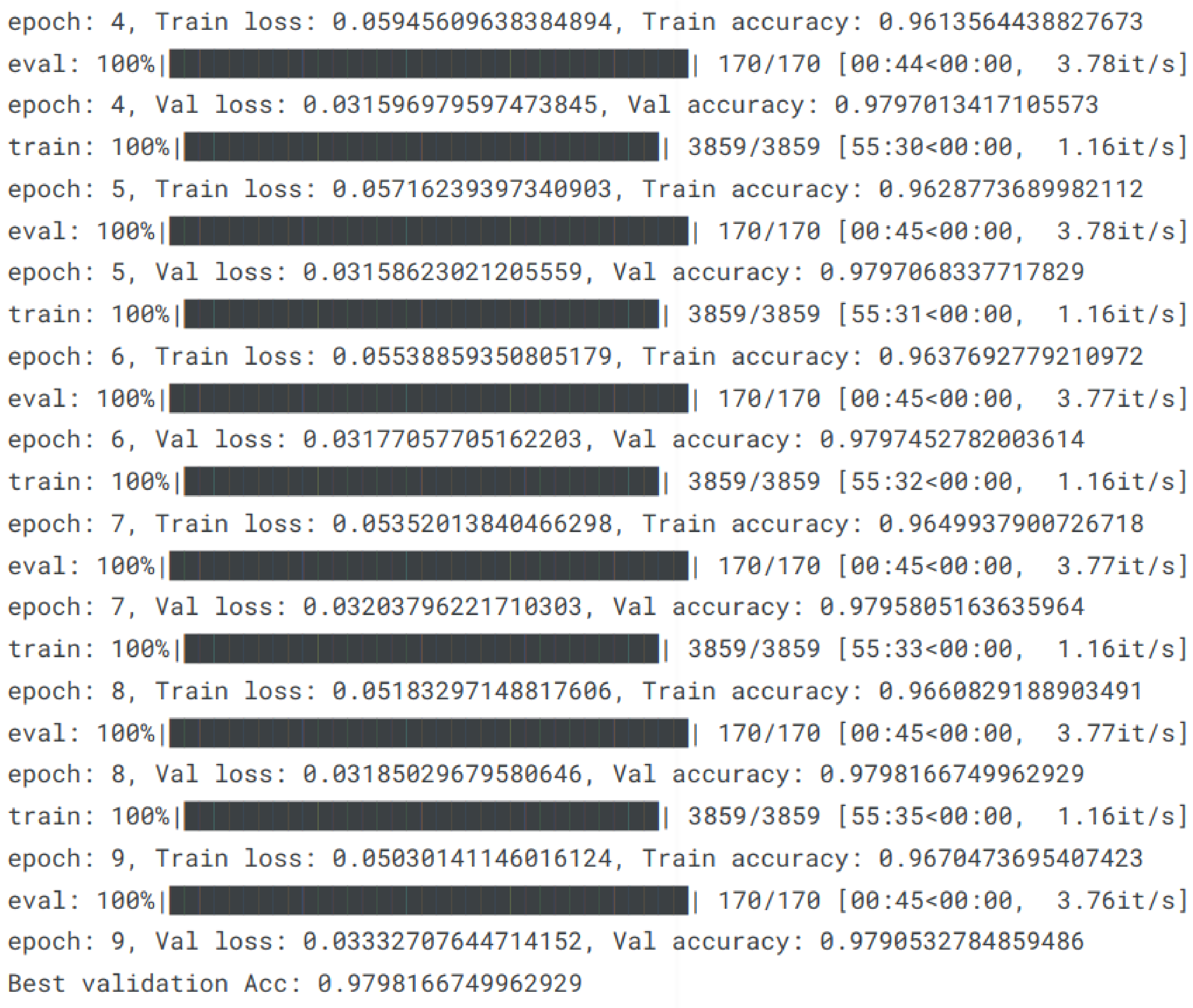
Additionally, the we explored the impact of substitution, insertion, and deletion augmentations on performance when applied separately. These augmentations are common techniques used to increase the robustness of machine learning models to variations in the input data. Specifically, we looked at the effect of substitution with a random token from the vocabulary, which means replacing a token in the input sequence with a randomly selected token from the vocabulary.It appears that the substitution augmentation did not have a significant impact on performance. However, it performed worse compared to substituting with the unknown token. We notice that the performance gain from different augmentations is larger on the ASR test set than on the reference test.

5.5 Discussion

In this section, we will discuss the challenges faced in developing language models for the Bangla language, which is relatively less studied and less resourced than English. One major challenge is that pre-trained monolingual language models for English tend to perform better than multilingual models, partly due to the larger amount of training data available for English. Another challenge is that the nature of the training data for Bangla can affect the performance of language models. In particular, the lack of punctuated transcribed data has led to the use of a news corpus for training, which is not ideal for capturing the nuances of spoken language, resulting in reduced prediction accuracy.

Moreover, the training data used for Bangla language models contains a mix of monologues and dialogues with varying degrees of complexity. This poses a challenge since different types of utterances may have different distributions of punctuation, such as periods and questions. For example, news data tends to have fewer questions and more periods, while dialogues often have more questions and shorter utterances. Therefore, it is important to consider the type of training data used in developing language models for Bangla and to ensure that the data is representative of the types of language used in real-world settings.

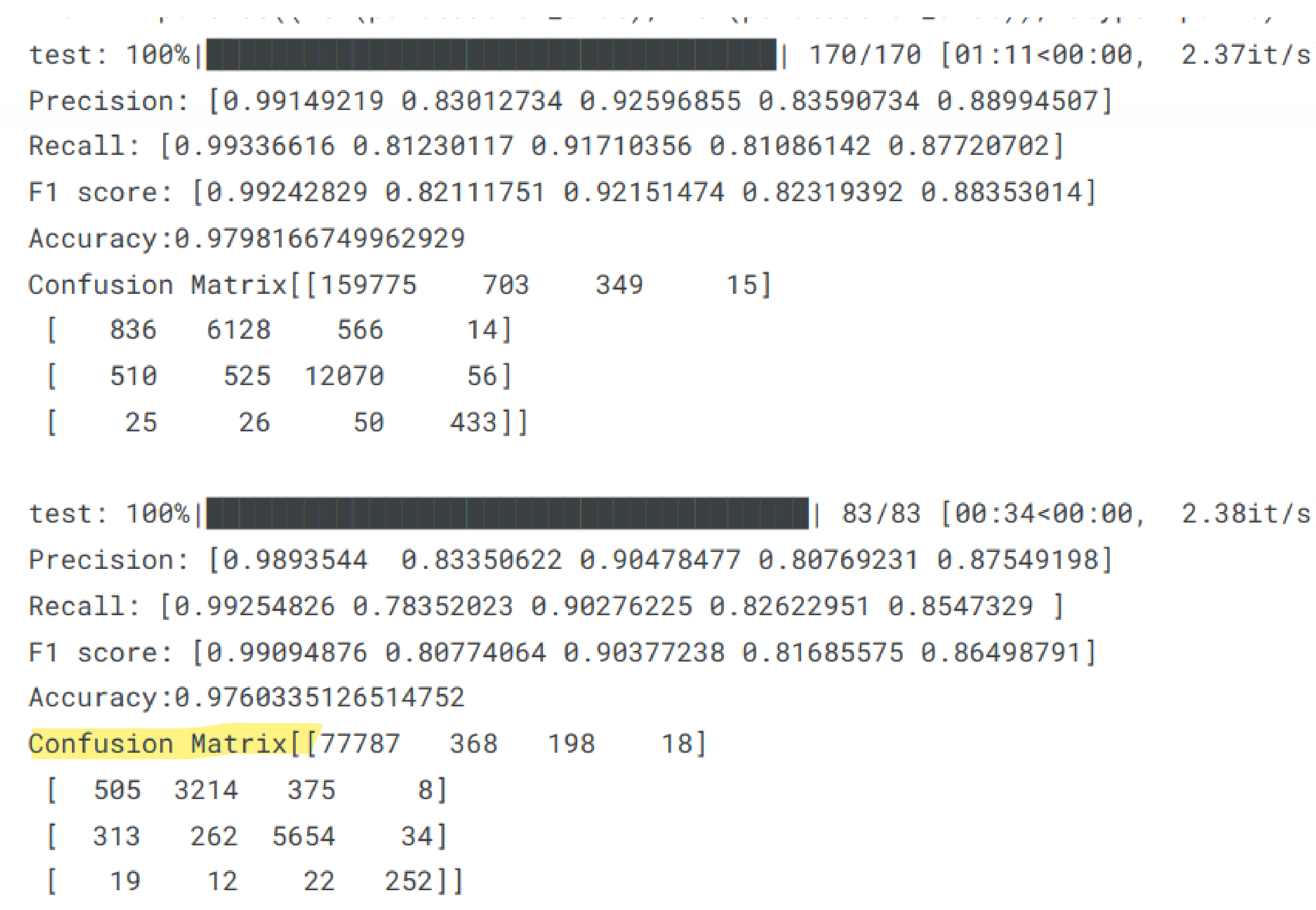
5.6 Training Epoch



```
epoch: 4, Train loss: 0.05945609638384894, Train accuracy: 0.9613564438827673
eval: 100%|██████████████████████████████████| 170/170 [00:44<00:00,  3.78it/s]
epoch: 4, Val loss: 0.031596979597473845, Val accuracy: 0.9797013417105573
train: 100%|███████████████████████████████| 3859/3859 [55:30<00:00,  1.16it/s]
epoch: 5, Train loss: 0.05716239397340903, Train accuracy: 0.9628773689982112
eval: 100%|██████████████████████████████████| 170/170 [00:45<00:00,  3.78it/s]
epoch: 5, Val loss: 0.03158623021205559, Val accuracy: 0.9797068337717829
train: 100%|███████████████████████████████| 3859/3859 [55:31<00:00,  1.16it/s]
epoch: 6, Train loss: 0.05538859350805179, Train accuracy: 0.9637692779210972
eval: 100%|██████████████████████████████████| 170/170 [00:45<00:00,  3.77it/s]
epoch: 6, Val loss: 0.03177057705162203, Val accuracy: 0.9797452782003614
train: 100%|███████████████████████████████| 3859/3859 [55:32<00:00,  1.16it/s]
epoch: 7, Train loss: 0.05352013840466298, Train accuracy: 0.9649937900726718
eval: 100%|██████████████████████████████████| 170/170 [00:45<00:00,  3.77it/s]
epoch: 7, Val loss: 0.03203796221710303, Val accuracy: 0.9795805163635964
train: 100%|███████████████████████████████| 3859/3859 [55:33<00:00,  1.16it/s]
epoch: 8, Train loss: 0.05183297148817606, Train accuracy: 0.9660829188903491
eval: 100%|██████████████████████████████████| 170/170 [00:45<00:00,  3.77it/s]
epoch: 8, Val loss: 0.03185029679580646, Val accuracy: 0.9798166749962929
train: 100%|███████████████████████████████| 3859/3859 [55:35<00:00,  1.16it/s]
epoch: 9, Train loss: 0.05030141146016124, Train accuracy: 0.9670473695407423
eval: 100%|██████████████████████████████████| 170/170 [00:45<00:00,  3.76it/s]
epoch: 9, Val loss: 0.03332707644714152, Val accuracy: 0.9790532784859486
Best validation Acc: 0.9798166749962929
```

Training Epoch

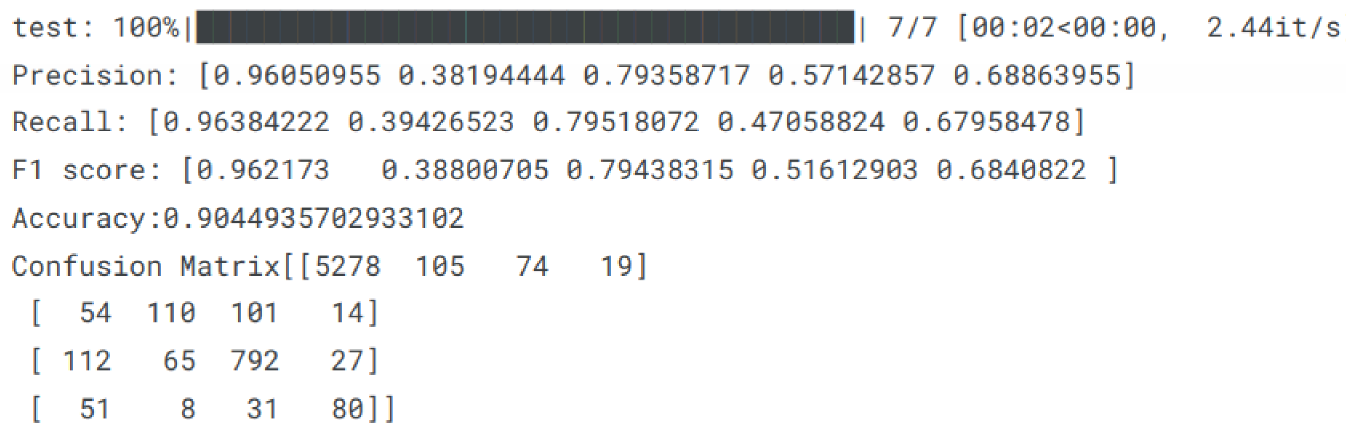
5.7 Result on the test dataset and Confusion Matrix



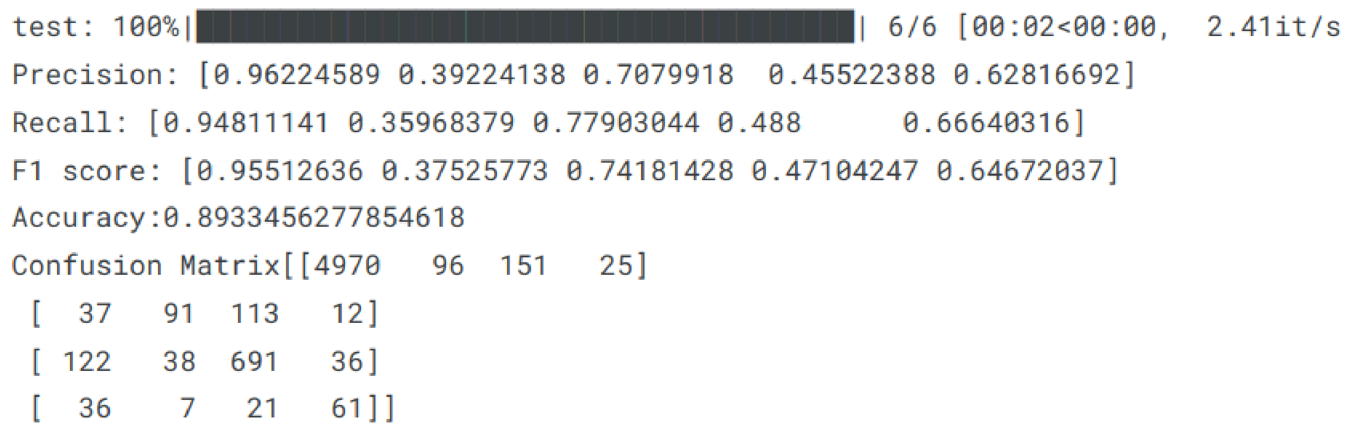
```
test: 100%|██████████████████████████████████████| 170/170 [01:11<00:00,  2.37it/s
Precision: [0.99149219 0.83012734 0.92596855 0.83590734 0.88994507]
Recall: [0.99336616 0.81230117 0.91710356 0.81086142 0.87720702]
F1 score: [0.99242829 0.82111751 0.92151474 0.82319392 0.88353014]
Accuracy:0.9798166749962929
Confusion Matrix[[159775    703    349     15]
 [   836   6128    566     14]
 [   510    525  12070     56]
 [    25     26     50    433]]

test: 100%|████████████████████████████████████████| 83/83 [00:34<00:00,  2.38it/s
Precision: [0.9893544  0.83350622 0.90478477 0.80769231 0.87549198]
Recall: [0.99254826 0.78352023 0.90276225 0.82622951 0.8547329 ]
F1 score: [0.99094876 0.80774064 0.90377238 0.81685575 0.86498791]
Accuracy:0.9760335126514752
Confusion Matrix[[77787   368   198    18]
 [  505  3214   375     8]
 [  313   262  5654    34]
 [   19    12    22   252]]
```

Result and Confusion matrix



```
test: 100%|████████████████████████████████████████| 7/7 [00:02<00:00,  2.44it/s
Precision: [0.96050955 0.38194444 0.79358717 0.57142857 0.68863955]
Recall: [0.96384222 0.39426523 0.79518072 0.47058824 0.67958478]
F1 score: [0.962173   0.38800705 0.79438315 0.51612903 0.6840822 ]
Accuracy:0.9044935702933102
Confusion Matrix[[5278  105   74   19]
 [  54  110  101   14]
 [ 112   65  792   27]
 [  51    8   31   80]]
```



```
test: 100%|████████████████████████████████████████| 6/6 [00:02<00:00,  2.41it/s
Precision: [0.96224589 0.39224138 0.7079918  0.45522388 0.62816692]
Recall: [0.94811141 0.35968379 0.77903044 0.488      0.66640316]
F1 score: [0.95512636 0.37525773 0.74181428 0.47104247 0.64672037]
Accuracy:0.8933456277854618
Confusion Matrix[[4970   96  151   25]
 [  37   91  113   12]
 [ 122   38  691   36]
 [  36    7   21   61]]
```

Result and Confusion matrix

Chapter 6

Future Work and Conclusion

Our study focuses on exploring transformer models for low-resource languages, specifically Bangla, and proposing a technique to improve punctuation restoration in noisy ASR texts. As there is limited research and resources available for Bangla punctuation restoration, our work is expected to make a significant contribution to the field. Our research outcomes and resources, including the created Bangla dataset and code, will be valuable for the research community and assist in further advancing the state-of-the-art for low-resource languages. By making the dataset and code publicly available, we hope to facilitate further research in this field and promote the development of more advanced models and techniques for punctuation restoration in low-resource languages.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Åukasz Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017, pp. 5998–6008. [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need

[2] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren, “Darpa timit acoustic-phonetic continuous speech corpus cd-rom,” in *Proceedings of the workshop on Speech and Natural Language*, 1993, pp. 274–278.

[3] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: An asr corpus based on public domain audio books,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[4] O. Klejch, P. Bell, and S. Renals, “Punctuated transcription of multi-genre broadcasts using acoustic and lexical approaches,” in *2016 IEEE Workshop on Spoken Language Technology*. United States: Institute of Electrical and Electronics Engineers (IEEE), Feb. 2017, pp. 433–440, 2016 IEEE Workshop on Spoken Language Technology, SLT 2016 ; Conference date: 13-12-2016 Through 16-12-2016. [Online]. Available: https://www2.securecms.com/SLT2016//Default.asp

[5] A. Vahdat, G. Kermia, and W. Byrne, “Sequence-to-sequence models for punctuated transcription combining lexical and acoustic features,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5700–5704. [Online]. Available: http://www.cstr.ed.ac.uk/downloads/publications/2017/icassp-2017.pdf

[6] F. Wang, W. Chen, Z. Yang, and B. Xu, "Self-attention based network for punctuation restoration," in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 2803–2808.

[7] J. Yi and J. Tao, "Self-attention based model for punctuation prediction using word and speech embeddings," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 7270–7274.

[8] J. Yi, J. Tao, Y. Bai, Z. Tian, and C. Fan, "Adversarial transfer learning for punctuation restoration," *CoRR*, vol. abs/2004.00248, 2020. [Online]. Available: https://arxiv.org/abs/2004.00248

[9] Q. Chen, M. Chen, B. Li, and W. Wang, "Controllable time-delay transformer for real-time punctuation prediction and disfluency detection," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8069–8073.

[10] T. Alam, A. Khan, and F. Alam, "Punctuation restoration using transformer models for high-and low-resource languages," in *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020). Online: Association for Computational Linguistics*, 2020, pp. 132–142. [Online]. Available: https://aclanthology.org/2020.wnut-1.18

[11] O. Tilk and T. Alumae, "Bidirectional recurrent neural network with attention mechanism for punctuation restoration," in *INTERSPEECH*, 2016.

[12] N. Shi, W. Wang, B. Wang, J. Li, X. Liu, and Z. Lin, "Incorporating external pos tagger for punctuation restoration," in *Proc. Interspeech 2021*, 2021, pp. 1987–1991.

[13] X. Che, C. Wang, H. Yang, and C. Meinel, "Punctuation prediction for unsegmented transcript based on word vector," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, 2016. [Online]. Available: https://www.aclweb.org/anthology/L16-1185.pdf

[14] G. Khan, K. Lee, S. Lee, H. Kim, and J. Kim, "Punctuation annotation using statistical prosody models," in *Proceedings of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03).* IEEE, 2003, pp. 612–615.

[15] K.-W. Lee, S. Lee, H. Kim, and J. Kim, "A combined punctuation generation and speech recognition system and its performance enhancement using prosody," *Computer Speech & Language*, vol. 21, no. 1, pp. 151–167, 2007. [Online]. Available: https://doi.org/10.1016/j.csl.2006.02.001

[16] M. T. Amin, M. S. Islam, and M. Z. Islam, "Leveraging a character, word and prosody triplet for an asr error robust and agglutination friendly punctuation approach," in *2021 International Conference on Bangla Speech and Language Processing (ICBSLP).* IEEE, 2021, pp. 1–5.

[17] N. Garg, A. Verma, and A. Khanna, "Self-attention based model for punctuation prediction using word and speech embeddings," in *2020 International Conference on Advances in Computing, Communications and Informatics (ICACCI).* IEEE, 2020, pp. 1313–1318.

[18] Y. Choi and C. Cardie, "Better punctuation prediction with dynamic conditional random fields," in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL '05).* Association for Computational Linguistics, 2005, pp. 290–297.

[19] A. Kumar, S. Kumar, and A. K. Ojha, "Punctuation prediction model for conversational speech," in *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT).* IEEE, 2019, pp. 1262–1267.

[20] A. Conneau, G. Lample, M. Ranzato, L. Denoyer, and H. JÃ©gou, "Unsupervised cross-lingual representation learning at scale," *arXiv preprint arXiv:1803.11175*, 2018.

[21] S. Gururani, M. Ma, A. Ali, S. Haddad, A. Abdelali, and N. Durrani, "Fast and accurate capitalization and punctuation for automatic speech recognition using transformer and chunk merging," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 7384–7388.

[22] T. Alam, A. Khan, and F. Alam, "Punctuation restoration using transformer models for high-and low-resource languages," in *Proceedings of the Sixth Workshop on Noisy*

User-generated Text (W-NUT 2020). Online: Association for Computational Linguistics, Nov. 2020, pp. 132–142. [Online]. Available: https://aclanthology.org/2020.wnut-1.18

[23] W. Gale and S. Parthasarathy, "Experiments in Character-Level Neural Network Models for Punctuation," in *Proc. Interspeech 2017*, 2017, pp. 2794–2798.

[24] JabRef Development Team, *JabRef*, 2016. [Online]. Available: http://www.jabref.org

[25] 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," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[26] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," *CoRR*, vol. abs/1909.11942, 2019. [Online]. Available: http://arxiv.org/abs/1909.11942

[27] I. Tenney, D. Das, and E. Pavlick, "Bert rediscovers the classical nlp pipeline," 2019. [Online]. Available: https://arxiv.org/abs/1905.05950

[28] J. Wei and K. Zou, "EDA: Easy data augmentation techniques for boosting performance on text classification tasks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).* Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6382–6388. [Online]. Available: https://aclanthology.org/D19-1670

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[30] E. Egonmwan and Y. Chali, "Transformer and seq2seq model for paraphrase generation," in *Proceedings of the 3rd Workshop on Neural Generation and Translation.* Hong Kong: Association for Computational Linguistics, Nov. 2019, pp. 249–255. [Online]. Available: https://aclanthology.org/D19-5627

[31] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014. [Online]. Available: http://arxiv.org/abs/1409.3215

[32] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014. [Online]. Available: http://arxiv.org/abs/1406.1078

[33] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[34] M. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *CoRR*, vol. abs/1508.04025, 2015. [Online]. Available: http://arxiv.org/abs/1508.04025

[35] Y. Zhang and Z. Rao, "n-bilstm: Bilstm with n-gram features for text classification," in *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, 2020, pp. 1056–1059.

[36] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," 2019. [Online]. Available: https://arxiv.org/abs/1909.13719