Improving Bengali and Hindi Large Language Models

by

Arif Shahriar

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science
University of Alberta

Abstract

Bengali and Hind are two widely spoken yet low-resource languages. The state-of-the-art in modeling such languages uses BERT and the Wordpiece tokenizer. We observed that the Wordpiece tokenizer often breaks words into meaningless tokens, failing to separate roots from affixes. Moreover, Wordpiece does not take into account fine-grained character-level information. We hypothesize that modeling fine-grained character-level information or interactions between roots and affixes helps with modeling highly inflected and morphologically complex languages such as Bengali and Hindi. We used BERT with two different tokenizers - Bengali and Hindi Unigram tokenizer and a character-level tokenizer and observed better performance. Then, we pre-trained two language models accordingly and evaluated them for masked token detection, both in correct and erroneous settings, across many NLU tasks. We provide experimental evidence that Unigram and character-level tokenizers lead to better pre-trained models for Bengali and Hindi, outperforming the previous state-of-the-art and BERT with Wordpiece vocabulary. We conduct the first study investigating the efficacy of different tokenization methods in modeling Bengali and Hindi.

Preface

This thesis is an original work by Arif Shahriar, done under the supervision of Dr. Denilson Barbosa. A paper by Arif Shahriar and Denilson Barbosa, describing this work has been submitted to The 2024 Joint Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024) and is currently under review. Another paper, by Arif Shahriar, Rohan Saha, and Denilson Barbosa, describing work by the candidate and another graduate student, under the supervision of Dr. Barbosa, titled "Relational Extraction on Wikipedia Tables using Convolutional and Memory Networks" has also been submitted to LREC-COLING 2024. That work is not part of this thesis. During an internship funded through an MITACS project, the candidate worked under the supervision of Dr. Barbosa toward building a statistical model of corrosion in industrial piping installations. A third paper, by Arif Shahriar, Eric Sjerve, Denilson Barbosa, describing that project is in preparation.

Acknowledgements

I would like to thank my supervisor, Dr. Denilson Barbosa, for his unwavering support, guidance, and patience throughout the process. His guidance and encouragement helped me learn how to conduct proper research and grow as a researcher. His constructive criticism helped me identify and address overlooked aspects, resulting in more meaningful work. Dr. Barbosa gave me complete freedom to choose my thesis topic and work on a problem that I am passionate about.

I would like to express my gratitude to Drs. Carrie Demmans and Greg Kondrak for spending their valuable time carefully reviewing my work and providing constructive feedback and revisions.

I would like to thank IRISNDT and the Department of Computing Science for funding my research and trusting me. I am grateful to Eric Sjerve from IRISNDT for providing me with this opportunity and his support, which has helped my research thrive.

I want to express my heartfelt gratitude to my beloved wife, Nazmun Nahar. She has always been my inspiration, motivator, and pillar of support during difficult times. Her constant encouragement has enabled me to overcome every challenge and transform it into an opportunity.

I would also like to thank my brother Fahim Faisal for igniting my passion for NLP research and inspiring me to explore new ideas. My parents have always been there for me, supporting my graduate studies and encouraging me to believe in myself.

Table of Contents

List of Tables

List of Figures

Abbreviations

ALBERT A lite BERT for self-supervised learning of language representations.

BBC British broadcasting corporation.

BERT Bidirectional encoder representations from transformers.

CLS Classification.

CNN Convolutional neural network.

CoCNN Coordinated convolutional neural network.

COPA Choice of Plausible Alternative.

ELECTRA Efficiently learning an encoder that classifies token replacements accurately.

ELMO Embeddings from language models.

EM Expectation maximization.

GB Gigabyte.

GLUE General language understanding evaluation.

GPT Generative pre-training.

GPU Graphics processing unit.

HOF Hate and offensive.

IIT Indian institute of technology.

IndicGLUE General language understanding evaluation for Indian languages..

IOB Inside–outside–beginning.

LOC Location.

MLM Masked language modeling.

NER Named entity recognition.

NFKC Normalization Form KC.

NLG Natural language generation.

NLI Natural language inference.

NLP Natural language processing.

NLU Natural language understanding.

NOT Non-hate and offensive.

NSP Next sentence prediction.

OBJ Object.

ORG Organization.

PER Person.

PPL Perplexity.

QA Question answering.

QC Question classification.

RAM Random Access Memory.

RoBERTa A robustly optimized BERT pre-training approach.

SEP Separator.

SOTA State of the art.

TB Terabyte.

XLM Cross-lingual masked language model.

Chapter 1

Introduction

Bengali and Hindi are the sixth and fourth most spoken languages globally, with 234 and 345 million native speakers, respectively [1]. Speakers of these languages contribute to 7% of the world's population. Bengali is the national language of Bangladesh, where 98% of the population have it as their first language. It is also the official language in two Indian states, West Bengal, Tripura, and the Barak Valley region of the state of Assam, and the second most spoken language out of the 22 official languages of India. Hindi is spoken mainly in northern India and is considered the common language of the Hind Belt region, covering most of India.

1.1 Motivation

Although Bengali and Hindi are widely spoken, computational resources are scarce for them. These languages have received very little attention and effort from the NLP community compared to high-resource languages like English, Chinese, and Spanish. Very few task-specific resources like labeled datasets and NLP tools are available for them. Therefore, self-supervised pre-training is the best way towards useful and large language models for such languages [2].

Self-supervised pre-training approaches can leverage factual knowledge and linguistic information from ample unlabeled raw texts to learn good word representations. These representations can improve downstream task performance with minimal

task-specific adaptation. The downstream tasks can be token-level like named entity recognition (NER) or sentence-level like sentiment analysis and natural language inference. Pre-training from such an unlabeled corpus can enable transfer learning to enhance natural language understanding capability, thus improving performance for downstream tasks with limited training data. Thus, it can benefit languages like Bengali and Hindi, which lack manually annotated data collection efforts.

Wikipedia and other unlabeled data resources are widely used for pre-training transformer-based language models. Like other languages, Bengali and Hindi Wikipedia and other web resources are continuously improving [2]. Through self-supervised pre-training, these languages can benefit from such unlabeled web resources.

1.2 Challenges

Bengali and Hindi belong to the Indo-Aryan language family. Both languages originated from Sanskrit [3] and share similar linguistic characteristics, including morphological richness. Modified vowels, consonants, and many compound characters lead to morphological complexity. When vowel letters follow a consonant, the shape of the vowel letter changes, and the modified vowel letter is placed to the left, right, both, or bottom of the consonant. Besides vowel letter attachment, consonants can take modified character forms when these appear on a compound character. While a vowel is not technically a letter but rather a sound, we refer to the letters that represent vowels in Bengali and Hindi.

A large number of compound characters are formed by joining multiple basic characters. Bengali has 50 basic characters, which can combine with each other and form up to 171 compound characters. Tables 1.1 and 1.2 present examples of modified vowels, consonants, and compound characters.

Bengali and Hindi are highly inflected languages. The same root can have many inflected forms depending on adding different suffixes and prefixes [4]. Table 1.3 shows examples of how various inflected words can be formed from each root.

Vowel letter	আ	ই	ঈ	উ	ঊ	এ	ঐ	ও	ঔ	ঋ
Modified vowel letter	া	ি	ী	ু	ূ	ে	ৈ	ো	ৌ	ৃ
Modified consonant	পা	পি	পী	পু	পূ	পে	পৈ	পো	পৌ	পৃ

Table 1.1: Vowel letters and their modified character forms. Modified vowel letters are added to one example consonant 'প'. Bengali has 11 letters that represent vowels, and 10 of these take modified forms when attached to any of the 39 consonants. Finally, if there are no modified vowel letters or '্' added to a consonant, it is assumed that the remaining vowel letter 'অ' follows the consonant 'প'.

Basic Characters	**Compound Character**
প্ + ত	প্ত
ষ্ + ক	ষ্ক
ল্ + ল	ল্ল
স্ + প্ + র	স্প্র
ন্ + দ্ + ব	ন্দ্ব
ক্ + ষ্ + ম	ক্ষ্ম

Table 1.2: Compound characters formed by joining multiple basic characters . Bengali has around 171 compound characters.

Root	**Inflected Forms**
হার ("hara'", to lose)	হারব ("hara'ba", will lose), হেরেছি ("herechhi", have lost), হারতাম ("hara'tAma'", used to lose)
ঘর ("ghara'", house)	ঘরে ("ghare", at home), ঘরগুলো ("ghara'gulo", houses), ঘরটি ("ghara'Ti", the house)
আমি ("Ami", I)	আমরা ("Ama'rA", we), আমার ("AmAra'", my), আমাকে ("AmAke", to me)

Table 1.3: Different inflected forms of same root leads to differnet words. Bengali has more than 160, 36, and 24 inflected variations of verbs, nouns, and pronouns.

Finally, these languages have a relatively free word order. Although these are head-final languages with principal subject-object-verb word order, word order is loosely bound in constituent chunks or local word group level. Nevertheless, intra-chunk

word re-ordering is not allowed [5].

The above characteristics make Bengali and Hindi different from high-resource languages like English, which require special attention for natural language tasks. Although modern English also has inflections, English is considered a weakly inflected language[1] with linear morphology [6]. However, Sanskrit-originated languages widely use nonlinearity in morphology [7], thus leading to nonlinear inflections. Inflections can be of two types depending on internal structural changes. Linear inflection indicates that the root structure remains unchanged while adding an affix. Nevertheless, euphonic changes can occur at the boundary. On the other hand, the internal structure of the root changes after being added to the affix in a nonlinear inflectional form. In Bengali, especially verbs show non-linear characteristics in inflectional form. For example, the verb root 'খা' ('kha' - to do) becomes 'খেয়েছি' ('kheyechi' - present perfect) after adding the suffix 'েয়ছি' ('echi'). This is an example of nonlinearity.

Pre-trained language representations from transformers [8] are the most popular choice for natural language tasks [9–11]. Current research focuses on better learning language representations from high-resource languages. State-of-the-art like BERT's [12] tokenizers are trained on high-resource languages. Even the multilingual and monolingual BERT-based language models [12, 13] inherited BERT's Wordpiece tokenization system for modeling Bengali and Hindi. Both systems rely on the original Wordpiece tokenizer, which, to the best of our knowledge, was not developed with these languages in mind. Therefore, the question remains: how suitable is the wordpiece tokenization system in the context of Bengali and Hindi? No prior work investigates the efficacy of different tokenization methods for modeling Sanskrit-originated languages. Our work thoroughly analyzed different tokenization methods to address the shared characteristics of Bengali and Hindi.

In fact, we observed that Wordpiece splits words into tokens that have no meaning in isolation and fails to separate roots, suffixes, and prefixes. Thus, the Wordpiece

[1]https://en.wikipedia.org/wiki/Inflection

tokenizer often fails to split words into natural tokens. Instead, it greedily selects the longest subword unit from the beginning and then repeats the same process until the end of the word. We found many situations where Wordpiece would pick a common root for its vocabulary during training while failing to use that token when encountering said root in a compounded and/or inflected form. One example was the root 'পরিবহন' ("paribahan'", transport); when Wordpiece was faced with 'গণপরিবহনের' ("gana'paribahan'er", of public transport), the resulting tokens would be sub-words ['গণপ', 'রি', 'বহ', 'নের'] that did not include a root, suffix, or prefix. We ran into many other examples where Wordpiece produced "unnatural" tokens.

Moreover, Wordpiece does not take into account fine-grained character-level information. Such information might help identify modified vowels, consonants, and compound characters, thus improving the capability of language models to deal with morphological complexity. Adding a single character, modified vowel, consonant, or compound character can produce a different word. Depending on the corpus, the Wordpiece tokenizer will tokenize it into other sub-word tokens, thus impacting the downstream task performance. For example, if we add a single character 'অ' at the beginning of the word 'কাজ' ("kaj", act), it will become 'অকাজ' ("Akaj", useless act). The single character added here is a prefix that indicates the negation of the word 'কাজ' ("kaj", act). The Wordpiece tokenizer will tokenize it into other meaningless subwords ['অক', 'াজ'] depending on the corpus, even if the meaningful subword unit 'কাজ' ("kaj", act) is present in its vocabulary.

1.3 Thesis Objectives

Motivated by Wordpiece's limitations observation, we carefully considered design choices for building NLU models for Bengali and Hindi, including evaluating different tokenization methods to accommodate the shared characteristics of these languages. Believing that better modeling fine-grained character-level information or interactions between roots and suffixes or prefixes would result in better models, we modified the

BERT architecture with two different tokenizers - Bengali and Hindi Unigram tokenizer and character-level tokenizer and observed better performance empirically. Bengali and Hindi Unigram tokenizer learns scores with vocabulary that help the tokenizer find the most probable splits of the word during segmentation. Therefore, it can better separate roots, affixes, and modified vowels. Thus, it might help the model learn how words are formed in Bengali and Hindi. Bengali and Hindi character-level tokenizers utilize CNN layers that attend to each character for constructing word embedding, thus better-identifying subword patterns and character-level interactions. Character-level interactions can be compound characters, modified vowels, and consonants. We pre-trained and evaluated language models based on these tokenizers on masked token detection, both in correct and in erroneous settings, among other common NLU tasks, including question classification, sequence labeling, text classification, and sequence pair classification.

The objectives of our thesis are summarized below.

1. Study Bengali and Hindi Unigram tokenizer, a fully probabilistic tokenizer that produces more meaningful subword units and is better equipped to deal with modified vowels and consonants, suffixes, and prefixes by finding the most likely split into tokens using scores learned during the tokenizer training process.

2. Study Bengali and Hindi character-level tokenizer based on CharacterBERT [14], which is well suited to deal with compound characters and better learn intra and inter-word patterns by consulting characters in each word.

3. Present two pre-trained models - Bengali and Hindi Unigram BERT and Bengali and Hindi Character BERT.

4. Provide experimental evidence that Unigram and character-level tokenizers lead to better pre-trained models for Bengali and Hindi and outperform BERT with Wordpiece vocabulary. Therefore, these two tokenizers seem to be better suited

for Bengali and Hindi than the original BERT tokenizer.

1.4 Thesis Outline

Chapter 1 provides motivation and background information regarding the research problem and challenges. In addition, the research question and objectives of the work are briefly discussed there. Chapter 2 discusses relevant literature and emphasizes areas where further study is required. Chapter 3 discusses different tokenizers and how they help achieve the thesis objectives. Chapter 4 presents the base model architectures and discusses how these language models are pre-trained. Moreover, this chapter explains why we selected certain datasets for pre-training and downstream tasks and the intrinsic and extrinsic evaluation methods used to assess model quality and performance. It also details downstream tasks with input, output, and description of datasets. Chapter 5 reports results, key findings, and a discussion. Chapter 6 re-emphasizes the thesis objectives and how introduced tokenizers and pre-trained models improve downstream task performance. This chapter also discusses potential future work to improve the models with different strategies. It concludes by suggesting how these tokenizers can be extended to other language models, such as language generation models. Finally, Chapter 7 discusses ethical considerations about pre-training data and the limitations of the tokenizers and pre-trained models.

Chapter 2

Literature Review

2.1 General Language Models

Careful design choices for language models (LM) help achieve better performance in high-resource settings [12, 14–17]. Transformer-based [8] models with self-attention mechanisms remain a state-of-the-art approach for language modeling. These models are trained using different pre-training objectives like auto-regressive language modeling, masked language modeling, and replaced token detection. Such models' success has shifted the attention from static word embeddings [18, 19] to contextual word representations obtained through pre-training.

GPT [15] adopted a generative pre-training objective to learn generalizable universal text representations from a large unlabeled corpus. The final aim of pre-training was to transfer learned knowledge to various downstream tasks. Their empirical results demonstrated that such an unsupervised pre-training strategy significantly improves performance in supervised target tasks. BERT [12] introduced a masked language modeling approach, improving GPT's auto-regressive pre-training technique. In BERT's masked language modeling method, a few input tokens were randomly masked, and the model learned by predicting the vocabulary utilizing context from the left and right directions. This bidirectional pre-training strategy enabled the model to produce deep bidirectional representations from a large corpus of unlabeled text. Moreover, they added an additional task-specific layer to fine-tune the model on

the target task. Instead of using unidirectional context, bidirectional representations facilitated the model to produce state-of-the-art results in eleven downstream NLP tasks.

RoBERTa [16] reimplemented the BERT model and pointed out overlooked design choices. The authors thoroughly examined the impact of hyperparameter tuning and the training set size and concluded that BERT was undertrained. To address this issue, they used more data for training with longer sequences and larger batch sizes. They also applied a dynamic masking strategy instead of static masking, where random masking patterns were generated each time data were fed to the model. Their suggested design choices resulted in improved performance on all downstream tasks. Moreover, their work re-established the effectiveness of the masked language modeling objective for natural language understanding.

Like RoBERTa, many other works introduced different variants of BERT [12]. El Boukkouri et al. proposed a variant named CharacterBERT [14]. CharacterBERT employed ELMO's [20] character-level CNN module to obtain word representation by consulting characters in sequence. Since they dropped the Wordpiece vocabulary of the BERT model, their approach had the advantage of having open vocabulary instead of being constrained to Wordpiece vocabulary. The purpose of CharacterBERT was mainly to deal with the vocabularies of specialized domains like the medical domain. The success of CharacterBERT in specialized domains motivates us to explore the advantage of the character-level system in the context of Bengali and Hindi, which include a lot of compound and modified characters.

Clark et al. [17] improve the BERT pre-training objective using a generator and discriminator model. Initially, the generator model corrupts a few tokens and replaces them with plausible alternatives instead of masking. After this, the discriminator model was trained to distinguish between original input and replaced input, allowing the model to learn faster from all tokens from the entire input sequence rather than a few masked tokens.

Such improvements motivate us to identify overlooked design choices for low-resource languages like Bengali and Hindi.

2.2 Multilingual Pre-trained Models

Most prior work focuses on high-resource languages like English, Chinese, and Spanish. Therefore, developing language models for low-resource languages like Bengali and Hindi remains understudied. The lack of specialized LMs for low-resource languages like Bengali and Hindi forces NLP researchers working on downstream tasks [21, 22] to resort to fine-tuning multilingual pre-trained language models (PLM). Delvin et al. released the multilingual BERT (mBERT) model [12]. It was pre-trained on a multilingual corpus obtained by concatenating Wikipedia pages encompassing 104 languages. It followed the transformer-based architecture of the original BERT model and was pre-trained using similar masked language modeling (MLM) and next sentence prediction (NSP) objectives. Their model had a shared Wordpiece vocabulary that covered the top 104 languages. Thus, the Wordpiece tokenizer was trained on all the languages.

A RoBERTa-based multilingual model, XLM-RoBERTa [23], was also pre-trained with MLM objective on more than 2 TB of filtered common crawl data encompassing 100 languages. They analyzed trade-offs between low and high-resource language and the effects of language sampling and vocabulary size. Their work showed that increasing model capacity can deal with the low-resource and high-resource language performance trade-offs and lead to superior performance on cross-lingual tasks. They used a vocabulary of 250K and trained two different models, XLM-R base and XLM-R large.

Finally, IndicBERT [24] is a multilingual PLM developed for 11 major Indian languages. They presented NLP resources for these 11 Indic languages. Their NLP resources comprise a large-scale general domain sentence-level monolingual corpora, an evaluation benchmark, and a PLM named IndicBERT. IndicBERT used ALBERT [25]

as a base model. However, the multilingual models cover a wide range of languages. They are usually larger models, requiring more computational cost to fine-tune for target tasks due to their larger shared vocabulary size and increased model capacity (1.6X - 2.5X parameters for mBERT, XML-R than BERT). We emphasize the lack of previous work for understanding language-specific needs and thus carefully choosing transformer architectures in the context of Bengali and Hindi to address those needs.

2.3 Language Specific Models

There are very few models specific to Bengali and Hindi. Recently, Bhattacharjee et al. [13] proposed a Bengali NLU model BanglaBERT based on BERT [12]. They pre-trained the model using ELECTRA's [17] replaced token detection objective on a 27.5GB corpus crawled from 110 popular websites. Moreover, they introduced another model jointly trained on Bengali and English corpus to enable cross-lingual zero-shot transfer. Their language-specific model performed better than multilingual models in supervised settings, demonstrating the importance of training language-specific models. Like other BERT-based models developed for high-resource languages, they trained a Wordpiece tokenizer for Bengali.

Moreover, Rahman et al. [4] have presented an analysis of different architectures like convolutional, recurrent, and transformer-based neural networks in the context of Bengali and Hindi. They proposed a memory-efficient Coordinated CNN (CoCNN) architecture for modeling Bengali and Hindi. CoCNN model utilized word and sentence-level convolutional submodules to learn intra-word and local sentence-level patterns.

Finally, they concluded that state-of-the-art transformer models could not perform better for Bengali and Hindi. However, their model with fewer parameters could outperform competitive architectures like state-of-the-art transformers. Although their work attempted to address the specific characteristics of Bengali and Hindi, the CoCNN model cannot be pre-trained. Instead, it needs to be trained end-to-end for

each downstream task. Therefore, it would not be able to transfer knowledge from large unlabeled corpora but only be limited to very few labeled datasets available for such low-resource languages.

To overcome such shortcomings, we utilize transformer architecture to enable transfer learning through pre-training and simultaneously address Bengali and Hindi language-specific needs.

Chapter 3

Tokenizers

Tokenizers heavily influence transformer-based language models because the text encoding method determines how the language model will perform in various language understanding tasks. Designing an LM for a language requires a lot of research to choose a tokenizer well suited for the specific language characteristics. The language model might not learn meaningful representations without understanding the language structure. We believe that finding the most meaningful word representations would lead to better language models. We observe that BERT's original Wordpiece tokenizer cannot produce state-of-the-art results for Bengali and Hindi. Thus, we propose modifications to that model using two different tokenizers - Bengali and Hindi Unigram tokenizer and character-level tokenizer, that improve end-task performance.

3.1 Wordpiece

A tokenizer is one of the essential components of natural language processing. Language models cannot directly process raw text, so converting raw text into numerical vectors is necessary. The purpose of the tokenizer is to split the text into pieces called tokens according to a set of rules or learned vocabulary and translate them to numerical values.

A tokenizer can be word level, sub-word level, or character level. Word-level tokenizers are built to split raw texts into words using spaces and pre-defined rules

such as punctuation. The word-level approach has disadvantages, such as extensive vocabulary size due to a larger corpus and a higher number of words in a language. Also, training a language model with an extensive vocabulary size becomes computationally expensive. Although the issue of extensive vocabulary size is solvable by removing less important words from vocabulary, this removal approach will result in many unknown words when encountering words not present in the vocabulary. In addition, such tokenizers will treat very similar words as different words, thus representing them using other numeric vectors or embeddings.

Subword-level tokenizers address the problem of word-based tokenizers by splitting words into smaller subwords. They can reduce vocabulary size to share information between different words. Thus, they can identify common subword units across very similar words. Subword-level tokenizers are better than word-level tokenizers since they can prevent the out-of-vocabulary word problem, use a manageable vocabulary size, and mitigate data sparsity by splitting words into common subword units [26].

Subword-level tokenizers are data-driven and rely on learning from the data. On the other hand, rule-based tokenizers rely on hand-crafted and language-dependent rules. For example, the SpaCy [27] library includes rule-based tokenizers that apply rules specific to language for tokenizing. They are word-based tokenizers, so they suffer from the same problem of larger vocabulary size faced by word-level tokenizers. Moreover, they are mainly designed for European languages where words are segmented using whitespaces [28]. Hence, languages such as Chinese that do not use spacing in writing systems can be challenging for such tokenizers [29]. NLTK library [30] offers a Penn tree bank word tokenizer [31] that uses regular expressions to tokenize text. Other examples of rule-based tokenizers include the Korean morphological analysis tool, which is used to create supervised tokenizers [29]. Moreover, language is always changing and evolving [32]. For example, the contemporary Bengali language evolved from Shadhu Bhasha. With the evolutionary change, verb morphology changed from a structured form in Shadhu Bhasha to a nonlinear form in today's Bengali (SCB)

[7]. Thus, designing new rules to cope with evolutionary language changes is laborious and expensive. Previous work has demonstrated that unsupervised data-driven tokenizers can outperform hand-crafted rule-based methods for machine translation [29].

Wordpiece [26] is a subword tokenization algorithm. It is developed for dealing with rare words during machine translation and is also used for BERT pre-training. The user specifies the desired vocabulary size. This size is usually selected based on model validation performance on downstream tasks and computation costs. Wordpiece algorithm starts with a small vocabulary and iteratively increases its vocabulary size to the desired number of tokens. The initial vocabulary consists of all the characters in the corpus, which are referred to as tokens. The algorithm works by learning merge rules to combine pairs of tokens until the desired vocabulary is reached. The algorithm learns these merge rules by combining two tokens of current vocabulary to create and add new tokens to the vocabulary. In the first few iterations, characters from the base vocabulary merge to create new tokens. Eventually, these newly added tokens merge and generate longer ones with subsequent training steps. Wordpiece computes scores for each of the pairs in the existing tokens and selects the one with the best score for merging. If two consecutive tokens in the pair are t_1 and t_2 and the merged pair is $t_1 t_2$, the score can be calculated as below.

$$Score = \frac{freq(t_1 t_2)}{freq(t_1) \times freq(t_2)} \tag{3.1}$$

Thus, the score can be calculated by dividing the pair frequency by the product of the frequencies of each token. This score emphasizes pairs where individual tokens are less frequent. Then, it selects the pair with the highest score for merging and repeats the same process until the desired vocabulary size is reached.

During tokenization, the Wordpiece tokenizer will identify the longest possible token at the beginning of a word. Once selecting the longest subword unit from the vocabulary, it will start again on the remaining part of the word until reaching the

end. For example, when tokenizing the word 'অকপটভাবে' ("AkapaT'bhabe", frankly), the Wordpiece tokenizer will first look for the longest token at the beginning of the word. Assume that 'অক' is the first identified longest token in the vocabulary. It will look for the longest token in the remaining split 'পটভাবে' and identify 'পট' as the longest token at the beginning of 'পটভাবে'. Finally, for the remaining part, 'ভাবে' is identified, which token is already in the vocabulary.

We do not discuss Byte-Pair Encoding tokenization (BPE) because Wordpiece is a variant of BPE. BPE [33] was initially developed as a text compression algorithm, and Radford et al. [15] first used this algorithm for tokenization while pre-training the GPT model. Wordpiece is very similar to BPE in terms of training, but it uses likelihood score instead of frequency for merging pair. Previous works have shown that Wordpiece performs better for neural machine translation tasks [26].

3.2 Unigram BERT

Unigram [28] is a subword tokenization algorithm. Unlike Wordpiece [26], it begins with a larger vocabulary and works in a top-down approach to iteratively reduce it to the final vocabulary. Before training the tokenizer, we apply normalization steps, including a few replacements and normalization form KC[1] (NFKC) Unicode normalization. NFKC Unicode normalization involves two steps – compatibility decomposition followed by canonical composition. Since some characters can be represented using different Unicode representations in Bengali and Hindi, NFKC normalization is used to normalize all variants of characters into a single shared form. This normalization step is also performed for the Wordpiece algorithm. Two or more whitespaces were replaced with a single space like SentencePiece [34] algorithm. We also preserve the accents since vowel matras and diacritics are frequently used in Bengali and Hindi.

We use a Metaspace pre-tokenizer to replace single whitespaces with a specific character ('_') for the pre-tokenization step. We start with an initial seed vocabulary

[1]https://unicode.org/reports/tr15/

(91,551) much larger than the target vocabulary size (30,522). Seed vocabulary includes all basic characters and the most common substrings (top 35%) obtained from the corpus. We include all possible characters besides the most common substrings because the model cannot otherwise tokenize potential out-of-vocabulary words. Out-of-vocabulary words occur when the tokenizer encounters words with subword tokens not present in the vocabulary.

At each iteration, the corpus-level loss for the Unigram algorithm was computed, given the current vocabulary. Each word in the corpus was tokenized using the available vocabulary for calculating loss. Like the Unigram language model, each token is considered independent of the previous tokens. Hence, each token's probability can be computed by dividing the token frequency by the sum of all tokens' frequencies in the corpus. During word tokenization, the Unigram model considers the best segmentation of the word into sub-word tokens with the highest probability. This best segmentation is efficiently done using the Viterbi [35] algorithm. As tokens are considered independent, word probability is the product of sub-word token probabilities. Afterward, word probabilities are multiplied by the word frequency to get final scores. Finally, the corpus-level loss is determined by applying the negative log-likelihood of these scores. Suppose that we have sample words $w_1, w_2, \ldots, w_n$ in the corpus; then the corpus-level loss can be computed as:

$$loss = \sum_{i=1}^{n} freq_i * (-log(P(w_i))) \tag{3.2}$$

The training is based on an expectation maximization (EM) algorithm. It determines how much the loss will be increased for removing each token from the current vocabulary. At each step, p% of the subword tokens having the most negligible impact on corpus-level loss are discarded. The p value for discarding tokens is a hyperparameter that was selected based on final corpus-level loss. These steps are repeated until the desired vocabulary size of 30.5K is reached. Moreover, the scores are saved with

vocabulary to find the most probable splits into tokens. Therefore, the algorithm preserved the tokens mostly needed for tokenization purposes and dropped less needed ones belonging to the tail end of the distribution. During training, elementary tokens are not discarded, as the model cannot tokenize every word without all possible characters. Since our NLU model is based on BERT, we used similar special tokens ([UNK], [PAD], [CLS], [SEP], [MASK]) to indicate unknown, padding, classification, separator, and mask tokens.

Reference word	**Unigram tokenizer**	**Wordpiece tokenizer**
শিক্ষাপ্রতিষ্ঠানেও (“shikKapRatiShThan'eO” , also in educational-institutions)	[শিক্ষাপ্রতিষ্ঠান, ে, ও] [(“shikKapRatiShThan'”, educational-institutions), (“e”, in), (“o”, also)]	[শিক্ষাপ্রতিষ্ঠানে, ও] [(“shikKpRatiShThan'e”, in educational-institutions) , (“o”, also)]
ভারোত্তোলনে (“bharottalan'e”, in weight-lifting)	[ভারোত্তোলন, ে] [(“bharottalan'”, weight -lifting), (“e”, in)]	[ভারোত্তোল, নে] [(-), (-)]
অকপটভাবে (“AkapaT'bhabe”, frankly)	[অকপট, ভাবে] (“AkapaT'”, frank), (“bhabe”, way)]	[অক, পট, ভাবে] [(_), (_), (“bhabe”, way)]
বিনম্রতা (“binamRa'ta”, modesty)	[বিনম্র, তা] [(“binamRa'”, humble) , (“ta”, being)]	[বিন, ম্, রত, া] [(_), (_), (_), (_)]
গণপরিবহনের (“gana'paribahan'er”, of public transport)	[গণপরিবহন, ের] [(“gana'paribahan'”, public transport), (“er”, of)]	[গণপ, রি, বহ, নের] [(_), (_), (_), (_)]
অকাজ (“Akaj”, useless act)	[অ, কাজ] [(“A”, useless), (“kaj”, act)]	[অক, াজ] [(_). (_)]

Table 3.1: Comparison of Unigram tokenizer with Wordpiece Tokenizer. (_) indicates the translation of a token without meaning that does not indicate any root, suffix, prefix, or meaningful unit.

Preliminary Experiments

We trained the Unigram tokenizer on Bengali pre-training data and compared it with the original BERT's Wordpiece tokenizer. We tokenized a few sample words and examined the differences (see Table 3.1).

Looking at the quality of sub-word tokens, we observed that the Unigram tokenizer always utilizes learned scores to find the most likely splits into tokens. It can separate roots (ভারোত্তোলন), emphasizing suffixes ('ও') and modified vowels ('ে') in the first and second examples. However, the Wordpiece tokenizer fails to do that. In the third and fourth examples, the Unigram tokenizer can separate the roots ('অকপট', 'বিনম্র') from suffixes ('ভাবে','তা') in their inflected forms. In addition, Unigram tokenizer can separate the affix ('ের') in the second last example. This affix produces an inflectional form of the noun to denote a relationship with the next word in a sentence. Nevertheless, the Wordpiece tokenizer cannot identify roots, suffixes, or prefixes during word segmentation. In the last example, the Unigram tokenizer produces meaningful sub-word tokens ('অ', 'কাজ'). Here, the first prefix ('অ') indicates negation, thus negating the meaning of the next token ('কাজ'). On the contrary, in the last three examples, the Wordpiece tokenizer unnecessarily decomposes words into meaningless word pieces ('অক', 'াজ'). These word pieces do not indicate any suffix, prefix, root, or meaningful unit. Therefore, these tokens might not help the BERT model understand how words can be formed in Bengali and Hindi.

Finally, we examined the number of roots and affixes covered in Wordpiece and Unigram vocabulary trained on Bengali pertaining corpus. For identifying roots and affixes, we used a list of 83,666 roots[2] and 62 inflectional affixes [7]. We observed Unigram has 31% more roots and 29% more affixes in its vocabulary compared to Wordpiece. Wordpiece vocabulary has 6,679 roots, whereas Unigram vocabulary has 8,757 roots. Moreover, Wordpiece vocabulary includes 45 affixes, whereas Unigram

[2]https://github.com/Foysal87/Bangla-NLP-Dataset/

vocabulary includes 58 affixes.

3.3 Bengali and Hindi Character BERT

The character-level tokenizer comprises a convolutional neural network (CNN) module that produces a contextless token-level representation before feeding to the BERT model. It takes as input a sequence of characters and embeds each character into a fixed-sized d-dimensional vector. These embeddings are sequentially fed into seven 1D CNN layers with various filter sizes. The outputs of CNN layers are max-pooled and concatenated to build token-level representations. Furthermore, these concatenated representations go through Highway layers [36] incorporating nonlinearities with residual connections. Finally, outputs are projected to 768 dimensions, similar to BERT's embedding size. These token-level embeddings are fed to BERT's 12 encoder layers to produce contextual representations. Figure 3.1 depicts how the CNN module constructs token-level representation after attending to each character in the sequence.

In Bengali and Hindi, characters can combine to form modified vowels, consonants, and compound characters. The character-level CNN module can model these intra-word interactions to learn fine-grained character-level information. Adding a single character, modified vowel, consonant, or compound character can lead to a different word. Thus, the original BERT's Wordpiece tokenizer will produce different sub-word tokens that can impact downstream task performance.

Moreover, the character-level tokenizer consults characters to produce a word-level representation. Therefore, it can have an open vocabulary not limited to Wordpiece vocabulary. Besides injecting character-level information, the Bengali and Hindi Character BERT model can learn local inter-word dependencies at the sentence level using encoder layers. Following the original BERT, we used padding, separator, classification, and mask tokens as special tokens and added positional embedding to the token embeddings. For MLM, we have constructed a temporary vocabulary compris-

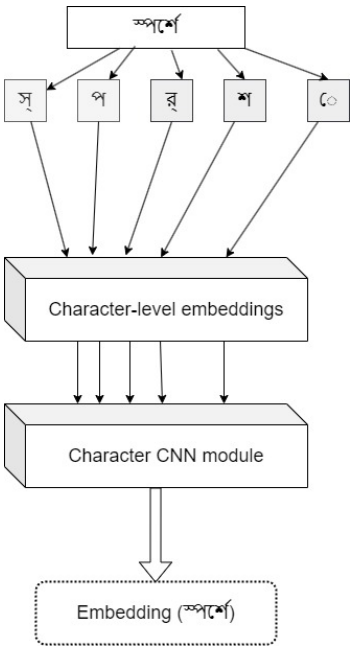

Figure 3.1: Illustration of Character CNN module producing a token-level embedding after attending each character. Modified vowels and compound characters are highlighted in Grey color.

ing the top 30,522 tokens from the pre-training corpus. These tokens have been used as target labels.

Chapter 4

Experimental Evaluation

4.1 Pre-training

4.1.1 Pre-training Objective

We pre-train the models with the masked language modeling (MLM) objective. During pre-training, 15% tokens are randomly selected for masking. The selected tokens are replaced using the [MASK] token. MLM cross-entropy loss was defined using the masked token prediction task. In addition, we randomize the masking pattern every time we feed a batch of sequences to the model. Random masking enables the model to learn efficiently from as many tokens as possible without duplicating data samples. We pre-trained the Bengali and Hindi Character BERT and Unigram BERT models on each language separately. However, we do not train models on the next-sentence prediction (NSP) task, as the authors of the RoBERTa [16] paper show that removing next-sentence prediction loss does not hurt the BERT model's performance on downstream tasks.

4.1.2 Model Architecture and Hyperparameters

Our models are based on the BERT base model and CharacterBERT model. Both models use a 12-layer encoder with 12 attention heads and 768 embedding sizes. We pre-trained the models for 803,640 steps with a 256 batch size. We use 128 maximum sequence length due to GPU memory constraints and pre-trained on a single A-100

GPU (40GB of VRAM). For optimization, we use the Adam optimizer [37] with a 2e-5 learning rate, L_2 weight decay of 0.01. We linearly decay the learning rate with training steps until the desired number of steps is reached. The Bengali and Hindi Unigram BERT model has 110M parameters. Moreover, the Bengali and Hindi Character BERT model has 104.6M parameters due to the CNN module's smaller character embeddings (16 dimensional).

4.2 Datasets

4.2.1 Pre-training Datasets

Pre-training transformer-based model requires a large quantity of good-quality data. For example, BERT used Wikipedia and Book Corpus [38] as pre-training data. We used open-access datasets [39] that represent the contemporary Bengali and Hindi writing styles. In addition, we select data sources that cover various topics. Many noisy and unauthentic data sources are available, including offensive text and erroneous facts. Therefore, most pre-training data were collected from sources such as online news portals, known as authentic data sources with minimum offensive texts.

We pre-trained on articles from online news portals and Wikipedia articles. Bengali pre-training data was collected from Prothom Alo (between 2014 and 2017) [40] and BDNews articles (between 2015 and 2017) [41]. Hindi pre-training corpus encompasses Wikipedia articles [42], Hindi Oscar corpus [43], HindiEnCorp 0.5 [44] dataset, WMT Hindi news crawl data [45]. Hindi has a comparatively larger collection of pre-training data than Bengali because it has more textual data available in digital format. Since it is a sentence-level corpus, all text passages and documents were split into sentences. The Bengali and Hindi pre-training corpus had 6.69M and 8.57M samples, respectively, with a maximum sequence length of 128 tokens after tokenization.

4.2.2 Datasets For Model Quality Analysis:

We experiment with five publicly available datasets [39] for model quality analysis – three Bengali and two Hindi datasets. We chose newspaper articles to verify the effectiveness of models across a wide range of domains. Domains include politics, technology, sports, lifestyle, literature, and entertainment. Bengali datasets include online news website articles collected from Prothom Alo [40], BDNews24 [41], and Naya Diganta [46]. The Hindi datasets are also from online news portals – Hindi News LiveHindustan [47] and Patrika [48]. Following Rahman et al. [39], we use two Bengali and Hindi datasets- Naya Diganta and Patrika as test sets.

We also tested our models with misspelled words. To do that, we replaced words in the Prothom Alo and Naya Diganta datasets with common misspellings identified by Rahman et al. [39]. The common misspellings are 20K real and synthetic error words formed from the top 13K words in the Prothom Alo dataset. First, we randomly select a certain percentage of sentences in the validation set. Then, we transform the words in the selected sentences with their misspelled versions according to identified common misspellings. For some of the words, multiple misspelled versions are available in the error words list. During word replacement, we arbitrarily select one misspelled version. We incrementally increased the noise ratio in the validation sets and created three versions with various noise levels. Dataset statistics are provided in table 4.1.

Corpus	**Training samples**	**Validation samples**	**Metric**
BDNews24	446,984	111,747	PPL
Prothom Alo	1,080,000	270,000	PPL
Naya Diganta	-	100,000	PPL
Livehindustan	187,077	46,770	PPL
Hindi Patrika	-	100,000	PPL

Table 4.1: Model quality analysis dataset statistics (the number of training and validation samples). Naya Diganta and Hindi Patrika were entirely used for validation purposes.

4.3 Performance Metrics

We use perplexity (PPL) obtained from the masked token prediction task to assess the quality of the model. Perplexity is a popular intrinsic evaluation metric that measures the language model's quality independent of downstream applications. To calculate the perplexity, we randomly select tokens from the sequences in the corpus for masking. Then, we compute the probability P of the model predicting token given the masked sequence. If we have probability values $P_1, P_2, \ldots, P_n$ for the corresponding tokens, we calculate perplexity by multiplying the probabilities and dividing by the number of tokens. We consider log of probabilities to avoid numerical underflow:

$$PPL = \exp(-\frac{1}{n}\sum_{i=1}^{n}(log(P_i))) \tag{4.1}$$

We ensure that the number of samples in the validation sets is sufficient for a good corpus representation. This PPL calculation approach is better suited to the BERT model than the next token prediction task because it measures the ability of the model to predict missing tokens anywhere in the sequence. This approach is similar to how BERT was designed and trained to leverage the bidirectional context to predict missing tokens at any position in the sequence.

For downstream tasks such as text classification, named entity recognition, sentiment analysis, and natural language inference (NLI), we use F1 score and accuracy as performance metrics.

4.4 Model Training & Evaluation

For the masked token prediction task, we trained models with the same vocabulary size (30,522) for 24 epochs. We used 80% of the data for training and reported perplexity on 20% of validation data. We optimized the models using Adam optimizer [37] with 2e-5 learning rate and decayed learning rate with linear learning rate scheduler. We used a batch size of 64 and trained the models using two GPU servers.

We used one server with a Tesla V100 GPU (16GB VRAM) and another with a Nvidia GeForce RTX 2080 TI GPU (11GB VRAM). For experimentation, we used frameworks and libraries such as Transformers [49], Pytorch [50], and Scikit-learn [51].

4.4.1 Fine-tuning Setup

We fine-tuned our pre-trained models for eight diverse downstream tasks in Bengali and Hindi. The fine-tuning was performed for 2-6 epochs, and the learning rate was tuned from 1e-5 to 5e-5 range with a weight decay of 0.01. We used the Adam optimizer and tuned learning rate warmup ratio from 0 to 10% of the total steps for model optimization. The batch size was chosen from {16, 32}, and we use a batch size of 32 for most of the tasks. The maximum length was restricted to 128 tokens for the fine-tuning experiments, as our models were pre-trained with a maximum sequence length of 128. We ran the experiments on two GPU servers with an Nvidia GeForce RTX 2080TI GPU (11GB VRAM) and a 3060 GPU (6GB VRAM), respectively. We fine-tune pre-trained models independently for each task.

4.4.2 Downstream Tasks

Few annotated datasets are available for Bengali and Hindi due to insufficient efforts in labeled data collection [2]. However, recent works [24, 39] have curated and unified existing datasets from publicly available sources and introduced NLU benchmark datasets for Bengali and Hindi. We fine-tuned models on Bengali and Hindi benchmark datasets [24, 39] comprising basic NLU tasks. Basic NLU task includes question classification, sequence labeling, text classification, and sequence pair classification. We provided detailed statistics of the datasets in table 4.2. Task details are discussed below.

Task	Corpus	Total	Train	Test	Dev	Metric
Question Classification	Bengali Question Classify	3,333	2,666	667	-	Macro-F1
Named Entity Recognition	Bangla NER dataset	71,285	64,155	3,565	3,565	Macro-F1
Genre Classification	Soham News Articles	14,106	11,284	1,411	1,411	Accuracy
	BBC News Articles	4,333	3,467	866	-	Accuracy
Hate Speech Detection	Hindi Hate Speech dataset	3,654	2,923	731	-	Macro-F1
Sentiment Analysis	Hindi Product Reviews	2,355	1,884	471	-	Macro-F1
	IITP Movie Reviews	3,100	2,480	310	310	Accuracy
Natural Language Inference (NLI)	COPA dataset	899	362	88	449	Accuracy
Discourse Analysis	MIDAS Discourse dataset	9,968	7,974	997	997	Accuracy
Question Answering	Cloze-style Multiple Choice QA	38,845	34,960	3,885	-	Accuracy

Table 4.2: Dataset statistics for eight diverse downstream tasks.

Question Classification

The Bengali Question Classify dataset [52] consists of 3,330 question samples and six classes. Bengali questions have flexible ways of inquiring, which poses difficulty for question classification (QC). The task is to classify the question into one of the six categories – numeric, human, location, abbreviation, entity, and descriptive type question.

Named Entity Recognition

We chose the largest openly available NER dataset for Bengali [53]. The dataset contains 71 thousand sentences properly annotated using the IOB tagging scheme. It was annotated using four course-grained tags: person (PER), location (LOC), organization (ORG), and object (OBJ) entity. The dataset consists of sentences from Bengali Wikipedia and three popular newspapers[1] – Ittefaq, Bangladesh Protidin, and Kaler Kontho. It encompasses domains such as politics, sports, health, entertainment, and editorial. Bangla NER poses unique challenges – no capitalizations to indicate named entity appearance, multiple meanings of the same word, no specific sentence structure for recognizing named entity patterns due to relatively free word order, inflected words leading to various surface forms of the same entity, and multiword expression leading to an entirely different meaning [21].

Article Genre Classification

A news article is provided as input, and the task is to classify the article's genre or topic. We select two Bengali and Hindi open-access datasets covering common article categories. Soham News Articles [54] is a collection of Bengali news articles (Anandabazar Patrika, Ebala, and Zeenews articles) with six genres – Kolkata, state, national, sports, entertainment, and international. In addition, BBC News [24] covers a wider range of genres, including the common ones - India, international, entertainment, sport, news, science, business, Pakistan, South Asia, Institutional, social, China, multimedia, and learning English.

Hate Speech Detection

The task is to identify whether a post on social media contains hate speech and offensive content. This task is a coarse-grained binary classification problem with two classes – Hate and offensive (HOF) and Non-hate and offensive (NOT). Hindi

[1]https://www.top10bd.com/top-10-newspaper-in-bangladesh

Hate Speech dataset [55] comprises 3,664 social media posts.

Sentiment Analysis

We used two Hindi datasets for sentiment analysis: Hindi Product Review [39] and IIT-Patna Movie Review [56] datasets. The product review dataset consists of 2,355 reviews. The task is to classify a review into positive or negative classes. For comparison with the baseline method, we consider positive and negative classes for reporting F1 score similar to previous work [4]. IIT-Patna movie (3,100 samples) reviews can be categorized into three polarities- positive, negative, or neutral. We report accuracy for the movie review dataset to compare with published results [24].

Natural Language Inference

For natural language inference (NLI), we use the Choice of Plausible Alternative (COPA) task [57]. This task verifies the models' capability for open-domain commonsense reasoning. We use COPA's Hindi-translated version [24] with 899 multiple-choice questions. The multiple-choice questions focus on casual reasoning about day-to-day activities with two possible choices. The question serves as a premise, and the objective is to choose an alternative with a more plausible causal relation (cause or effect) to the premise.

Discourse Analysis

The MIDAS Discourse dataset [58] comprises sentences collected from Hindi short stories and is annotated with five discourse modes - descriptive, narrative, dialogue, argumentative, informative, and others. The task is to identify the modes of discourse at the sentence level. Descriptive sentences indicate locations in stories. A narrative sentence denotes an action an entity performs and its association with the story timeline. Dialogue sentences express conversations, and argumentative sentence validates a claim. Finally, an informative sentence informs readers about an entity or situation in the story.

Cloze-style Multiple Choice QA

Cloze-style multiple choice question answering (QA) [24] evaluates whether a language model can serve the purpose of a knowledge base. In this dataset, 38,845 article samples are collected from Bengali Wikipedia. The entities in the text are recognized with their type using Wikidata. An entity is randomly masked in the article. The task is to predict the masked entity out of four possible candidates. The Cloze-style QA task is challenging because three incorrect entities also appear in the article and belong to the same entity type as the correct one.

Input Output Representation

We utilize task-specific input-output representations for fine-tuning Bengali and Hindi Character BERT and Unigram BERT on downstream tasks. For sequence labeling tasks such as NER, we feed each sequence to the models and obtain final token representations. We assign labels to each token and use multiclass cross-entropy loss for finetuning.

Sequence classification includes QC, genre classification, sentiment analysis, discourse analysis, and hate speech detection tasks. For sequence classification, we feed the final hidden vector aggregate representation of the classification token ([CLS]) from the last layer to a linear classifier with a softmax layer to obtain the probability of each class. We used a similar approach for the NLI task, providing a sequence pair separated by a separator token ([SEP]) into the model instead of a single sequence. The classification token ([CLS]) representation is fed into the output layer for classification into one of the categories.

We treat cloze-style multiple-choice QA as a multiple-choice task. We input the masked article and candidate entity separated by a separator ([SEP]) token into a classifier. The classifier predicts the candidate entity with the highest score among the four options. We used the correct entity index number as a label out of all probable choices and fine-tuned the model using cross-entropy loss.

4.4.3 Evaluation

We select the best model based on validation performance for public datasets with given test and validation splits. Finally, we load the best model and use the official test set to report the final result. We perform a five-fold cross-validation for the data sets without an official test set and report mean results. We repeat the whole process for each task.

We have computed measures of dispersion for our reported results. For five-fold cross-validation, we report the standard deviation on the mean results obtained from five folds.

For datasets with an official test split, the standard deviation is calculated using a bootstrapping procedure. This involves randomly selecting samples from the annotated test set and obtaining a performance score. We repeat this process three times and report the results along with the standard deviation.

Chapter 5

Results and Discussion

5.1 Comparison With Baseline Model

We trained original BERT and proposed models on five Bengali and Hindi datasets. We compared their perplexity (PPL) scores on a validation set in correct and erroneous settings. Perplexity and improvements are reported in tables 5.1, 5.2, 5.3 and 5.4. We report improvement relative to the baseline. The PPL scores in the table show that the Bengali and Hindi Unigram BERT and Character BERT outperform baseline BERT with Wordpiece vocabulary. Bengali and Hindi Unigram BERT significantly improved over the baseline, and Character BERT also shows relatively better PPL scores.

We use eleven sets of perplexity values from Bengali and Hindi Unigram BERT and baseline BERT to perform the Wilcoxon Signed-Ranks test. Wilcoxon Signed-Ranks test is a non-parametric counterpart of paired t-test, which does not require data to be normally distributed. We compute the difference between sets of scores obtained from two models. The differences are ranked, and signs (positive/negative) of differences are also considered. Finally, the sum of the ranks is calculated for positive and negative ranks. The sum of positive and negative ranks should be the same if there are no differences. The null hypothesis is that the central tendencies of Bengali and Hindi Unigram BERT and baseline BERT are the same. We obtained an asymptotic significance value of 0.003, which is less than the significance level of 0.05. Hence,

we can reject the null hypothesis and conclude that there is a statistically significant difference between Bengali and Hindi Unigram BERT and baseline BERT PPL scores. We performed the same test for Bengali and Hindi Character BERT and baseline BERT and found PPL score difference was statistically significant (asymptotic significance value of 0.003).

Dataset	**Error**	**Wordpiece Tokenizer**	**Unigram Tokenizer (Bengali)**	**Improvement**
BDNews	No	100±0.7	**49±0.5**	+51%
Prothom Alo	No	45±0.3	**26±0.2**	+42%
	10%	53±0.3	**29±0.2**	+45%
	20%	62±0.2	**33±0.1**	+47%
	30%	72±0.1	**37±0.1**	+49%
Naya Diganta	No	81±0.5	**46±0.4**	+43%
	10%	95±0.4	**51±0.4**	+46%
	20%	110±0.4	**58±0.3**	+47%
	30%	126±0.8	**63±0.8**	+50%

Table 5.1: Improvement resulted from Bengali Unigram BERT compared to baseline BERT. Perplexity values are used to perform a Wilcoxon Signed-Ranks test, and results in bold text indicate a statistically significant difference in PPL score ($p < 0.05$).

Dataset	**Error**	**Wordpiece Tokenizer**	**Unigram Tokenizer (Hindi)**	**Improvement**
Live Hindustan	No	57±0.5	**23±0.3**	+60%
Hindi Patrika	No	92±0.3	**36±0.1**	+61%

Table 5.2: Improvement resulted from Hindi Unigram BERT compared to baseline BERT. Perplexity values are used to perform a Wilcoxon Signed-Ranks test, and results in bold text indicate a statistically significant difference in PPL score ($p < 0.05$).

Dataset	Error	Wordpiece Tokenizer	Character BERT (Bengali)	Improvement
BDNews	No	100±0.7	**74±0.1**	+26%
Prothom Alo	No	45±0.3	**35±0.1**	+22%
	10%	53±0.3	**36±1.0**	+32%
	20%	62±0.2	**38±0.1**	+39%
	30%	72±0.1	**40±0.2**	+44%
Naya Diganta	No	81±0.5	**49±0.1**	+40%
	10%	95±0.4	**51±0.1**	+46%
	20%	110±0.4	**54±0.5**	+51%
	30%	126±0.8	**57±0.5**	+55%

Table 5.3: Improvement resulted from Bengali Character BERT compared to baseline BERT. Perplexity values are used to perform a Wilcoxon Signed-Ranks test, and results in bold text indicate a statistically significant difference in PPL score ($p < 0.05$).

Dataset	Error	Wordpiece Tokenizer	Character BERT (Hindi)	Improvement
Live Hindustan	No	57±0.5	**43±0.1**	+25%
Hindi Patrika	No	92±0.3	**56±0.3**	+39%

Table 5.4: Improvement resulted from Hindi Character BERT compared to baseline BERT. Perplexity values are used to perform a Wilcoxon Signed-Ranks test, and results in bold text indicate a statistically significant difference in PPL score ($p < 0.05$).

We also experimented with whether Bengali and Hindi Character BERT and Unigram BERT can deal with misspellings. Hence, we created noisy versions of the Prothom Alo and Naya Diganta validation sets with incremental changes in noise levels. The experiments for verifying the robustness of models to misspellings are conducted only on validation sets because language models' performance often deteriorates in practical use in erroneous settings. We replace words in $p\%$ of the sentences

with common misspellings to create erroneous versions of the validation sets. The misspelled words are formed by adding, removing, replacing characters, modifying vowels and consonants, or producing inflected forms of words by adding affixes.

With incremental changes in error percentage, the perplexity of Bengali and Hindi Unigram BERT and Character BERT increases slowly compared to the baseline BERT. In particular, Bengali and Hindi Character BERT outperform original BERT by a large margin in erroneous settings. Although the relative improvement is 22% in the correct Prothom Alo validation set, the improvement becomes 44% when a noise level of 30% is applied to the Prothom Alo validation set. So, the relative improvement almost doubles, showing Bengali and Hindi Character BERT's advantage in erroneous settings. Similarly, Bengali and Hindi Character BERT shows robustness to misspellings in the Naya Diganta validation set, as the relative improvement increases by 15 points in most erroneous settings. Since the character-level tokenizer attends to each character in words, corrupting a word by modifying characters can have the least impact. The character-level tokenizer might still identify the intra-word patterns by consulting the correct characters in words. On the other hand, the Wordpiece tokenizer produces different subword tokens for a misspelled word, thus drastically impacting the model's performance.

Moreover, Bengali and Hindi Unigram BERT can also better adapt to noisy and inflected settings compared to the baseline model, as the relative improvement increases to 49% in the noisiest version from 42% in the correct Prothom Alo validation set. Similarly, Bengali and Hindi Unigram BERT show a 7-point increase in relative improvement for the Naya Diaganta validation set. It indicates that the Bengali and Hindi Unigram tokenizer's ability to separate modified vowels, affixes, and roots remains an advantage of the model to better adapt with misspelled and inflected words than baseline BERT.

We also plot the validation loss of models in figure 5.1, 5.2, and 5.3.

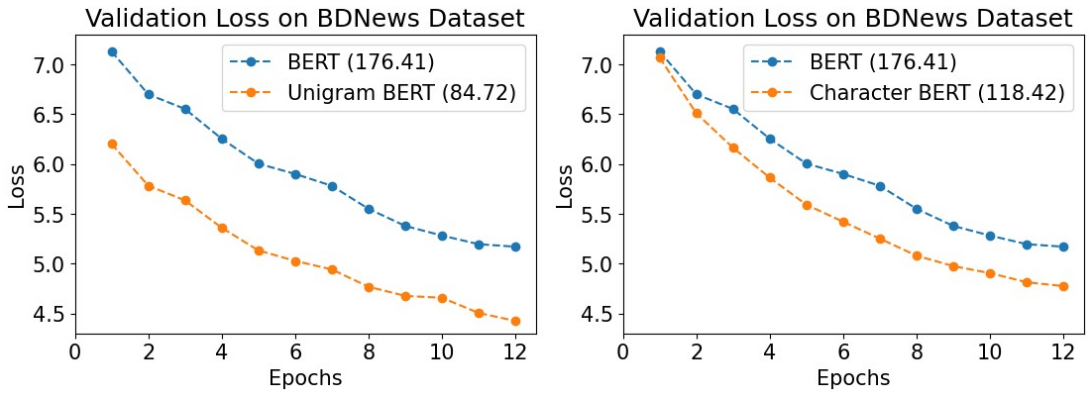

Figure 5.1: Comparing BERT with Bengali Unigram BERT and Character BERT on BDNews validation set for first 12 epochs. The perplexity after 12 epochs is shown with each model name.

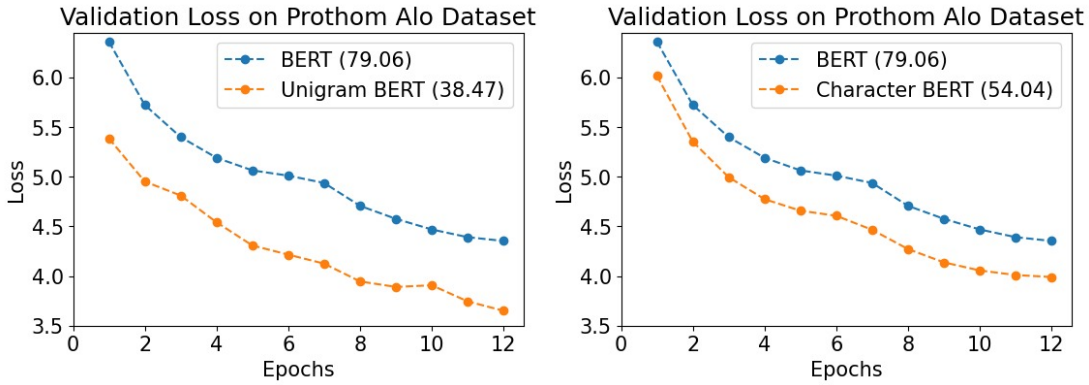

Figure 5.2: Comparing BERT with Bengali Unigram BERT and Character BERT on Prothom Alo validation set for first 12 epochs. The perplexity after 12 epochs is shown with each model name.

5.2 Comparison in Downstream Tasks

5.2.1 Comparison with Monolingual Model

We fine-tuned pre-trained Bengali and Hindi Character BERT and Unigram BERT on three downstream tasks. We compared them with the original BERT pre-trained on the same pre-training data [4]. There are no publicly available standard test splits for question classification, hate speech detection, and product review datasets. Thus, we perform 5-fold cross-validation to report mean F1 scores. The mean results in table 5.5 and 5.6 show that Bengali and Hindi Character BERT and Unigram BERT

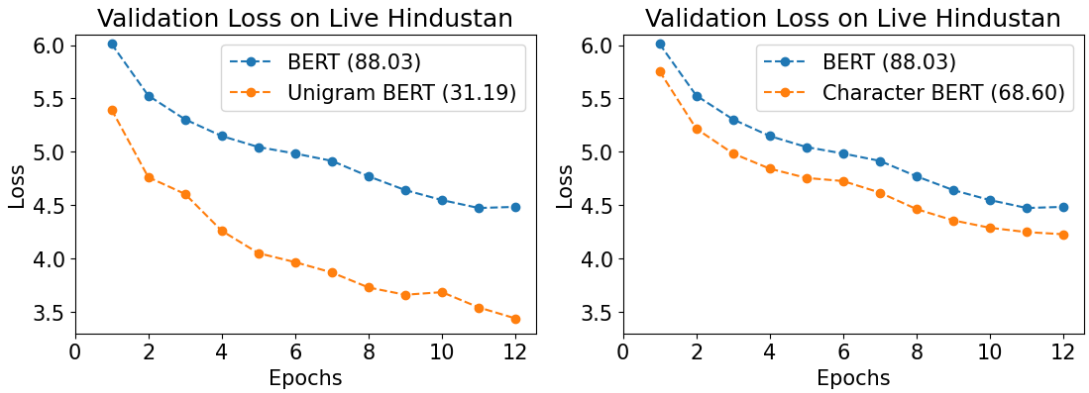

Figure 5.3: Comparing BERT with Hindi Unigram BERT and Character BERT on Live Hindustan validation set for first 12 epochs. The perplexity after 12 epochs is shown with each model name.

achieve robust performance over the original BERT in three downstream tasks. The Bengali and Hindi Unigram BERT model significantly improved over the original BERT. Moreover, character-aware Bengali and Hindi Character BERT perform at the same level as Unigram BERT on hate speech detection. However, it marginally lags behind Bengali and Hindi Unigram BERT in the question classification and product reviews tasks.

Dataset	**Original BERT**	**Bengali Unigram BERT**	**Bengali Character BERT**
Question Classify	90.50	**97.22**$\pm$**0.9**	96.48$\pm$0.4

Table 5.5: Comparison of F1 score between proposed pre-trained models and original BERT in Bengali downstream tasks. Original BERT results are from [4].

We do not re-train the original BERT and reproduce the baseline BERT result because it is computationally expensive to pre-train a BERT model from scratch. Therefore, we compare with published BERT results from [4], which uses the same dataset for pre-training.

Dataset	Original BERT	Hindi Unigram BERT	Hindi Character BERT
Hate Speech Detection	77.00	**82.93±0.6**	82.77±1.3
Product Reviews	84.10	**89.57±1.8**	87.23±1.9
Average	80.55	**86.25**	85.00

Table 5.6: Comparison of F1 score between proposed pre-trained models and original BERT in Hindi downstream tasks. Original BERT results are from [4].

Dataset	Indic BERT	Multilingual BERT	Bengali Unigram BERT	Bengali Character BERT
Named Entity Recognition	62.42	64.54	**71.54±0.1**	69.87±0.2
Soham News Article	78.45	80.23	91.29±0.1	**91.93±0.1**
Cloze-style QA	39.40	36.23	**56.13±0.1**	40.51±0.1
Average	60.09	60.33	**72.99**	67.44

Table 5.7: Accuracy comparison between proposed pre-trained models and multilingual models in Bengali downstream tasks, except NER task, which compares F1 score. NER result and the rest of the results for multilingual BERT and IndicBERT are published in [21] and [24], respectively.

5.2.2 Comparison with Multilingual Models

We compare our proposed pre-trained models with two multilingual models. Multilingual BERT represents the original BERT [12] pre-trained on multilingual corpora. IndicBERT is a pre-trained multilingual model [24] for Indian languages. Table 5.7 and 5.8 compares published multilingual BERT and IndicBERT [21, 24] results with our pre-trained models. However, the gold labels for the COPA test set are not publicly available. Therefore, we collected the test set with gold labels from the English COPA dataset [59]. Then, we translated them to Hindi using manually translated annotations [60]. We fine-tuned pre-trained multilingual BERT and IndicBERT on

Dataset	Indic BERT	Multilingual BERT	Hindi Unigram BERT	Hindi Character BERT
BBC News Classification	74.60	60.55	**76.67**±0.2	76.11±0.1
IITP Movie Reviews	59.03	56.77	**67.03**±0.3	64.18±0.4
Midas Discourse	78.44	71.20	**81.42**±0.1	79.34±0.1
COPA	51.22	54.78	**60.84**±0.6	56.23±0.1
Average	65.82	60.83	**71.49**	68.97

Table 5.8: Accuracy comparison between proposed pre-trained models and multilingual models in Hindi downstream tasks. The results for multilingual BERT and IndicBERT are published in [21] and [24], respectively.

the COPA dataset and reported results on the translated test set. Previous work [21] only published multilingual BERT results for the NER task. Hence, we fine-tuned IndicBERT on the NER dataset and reported results on the standard test set. We report macro-F1 for the NER task and accuracy for the other tasks.

Bengali and Hindi Unigram BERT outperform multilingual BERT and IndicBERT, achieving average scores of 72.99 and 71.49 in Bengali and Hindi datasets, respectively. Bengali and Hindi Character BERT outperform multilingual BERT and IndicBERT, achieving average scores of 67.44 and 68.97 in Bengali and Hindi datasets, respectively. Our pre-trained models' performance is more robust for four text classification tasks than for other tasks. Text classification tasks include Soham News Article, BBC News Article, IITP Movie Reviews, and Midas Discourse datasets. In sequence labeling tasks such as NER, Bengali and Hindi Character BERT and Unigram BERT exceeds multilingual models. For challenging tasks like COPA and Cloze-style QA, our models, especially Bengali and Hindi Unigram BERT, improve by a large margin of up to 9 points and 19 points in COPA and QA datasets, respectively. For

Wikipedia-based datasets like Clozed-style QA, our pre-trained models can improve task performance over multilingual BERT pre-trained on the Wikipedia corpus. Bengali and Hindi Character BERT and Unigram BERT are comparable, but Unigram BERT is consistently better in most tasks.

5.3 Error Analysis

For question classification, our pre-trained models make the highest errors for classifying samples in the "description" category. Descriptive questions typically ask about the definition, reason, and description of something. A descriptive question example is "আলীনগর সন্ধিতে নবাব সিরাজউদ্দৌলার সম্মতির অন্যতম কারণ কোনটি?" ('Which was one of the reasons for Nawab Sirajuddaula's consent to the Treaty of Alinagar?'). These questions (7% of the "description" category samples) ask about specific entities. Therefore, the models misclassified these questions as belonging to the 'entity' class. On the other hand, questions from entity class query about substance, symbol, currency, a particular term, language, animal, sport, technique, event, religion, disease, creative content, and others. For example, "গিয়াসউদ্দনি আজম শাহের পৃষ্ঠপোষকতায় শাহ মুহম্মদ সগীর কোন কাব্যটি রচনা করেন?" ('Which poem was written by Shah Muhammad Sagir under the patronage of Ghiyasuddin Azam Shah?') is an example question belonging to the entity class. Moreover, the 'description' class has the least samples compared to the other classes. Hence, the models do not see sufficient examples to discern descriptive questions from entity-related questions.

For hate speech detection, our models demonstrate lower precision for the 'non-hate and offensive' (NOT) class than the 'hate and offensive' (HOF) class despite having almost equal samples for both classes. Approximately 54% of the observations belong to the HOF class, and the rest (46%) belong to the NOT class. After inspecting misclassified observations, we notice that misclassified samples have frequently occurring English words ('anniversary', 'drink', 'female', 'twitter', 'force') as these observations are collected from social media posts. However, our models are pre-trained mainly

on monolingual newspaper data, which does not usually contain mixed scripts such as Hindi and English. For instance, our Bengali and Hindi Unigram BERT does not have all the English alphabets in its vocabulary, thus leading to [UNK] tokens for out-of-vocabulary alphabets. Although Bengali and Hindi Character BERT does not rely on a fixed vocabulary, it has not seen such mixed scripts during pre-training. This can impact the precision score, therefore impacting F1 for the NOT class. This issue might be solved by adding frequently occurring English words or all English alphabets to the model's vocabulary and retraining the models on a larger mixed script dataset.

For product reviews, both models yielded lower recall scores for the negative class, which impacts overall macro F1. The product reviews dataset consists of fewer negative examples than positive ones. Therefore, the smaller amounts of samples were not adequate for the models to learn accurately about the negative class during fine-tuning. We observed a similar issue in the movie review dataset, where most misclassified examples belong to the negative or neutral class.

In the NER task, words with multiple meanings are the main challenge for Bengali and Hindi Character BERT and Unigram BERT. For example, our models identified the word 'বাংলাদেশ' (Bangladesh) as the beginning of a location (B-LOC). This word itself is the name of a country, which is why the model misclassified it as location. However, it appears as 'বাংলাদেশ ইউনিফাইড দল' ("Bangladesh Unified dala'", Bangladesh Unified Team), the name of a sports team representing Bangladesh in the given context. Therefore, the correct label should be the beginning of an organization (B-ORG). Similarly, the model predicts 'শান্ত' ("Shanta'", quiet) as the beginning of a person entity (B-PER), which is wrong. In this context, the word means quiet, which should be assigned to the other tag (O). The model confused others with the person entity because it is common to name people 'শান্ত'. The second challenge for the models was to deal with multiword expressions. For example, the word 'উত্তর-পশ্চিমাঞ্চলের' ("Uttara'-poshcimaNcal'er", north-western region) is a multiword expression in mis-

classified samples. Bengali and Hindi Unigram BERT was partially correct as it successfully identified the first part as the beginning of a location (B-LOC) but misclassified the second part as other (O) type, which is also a location. The reason could be that the dataset has few examples of multiword expressions. On the contrary, character-aware Bengali and Hindi Character BERT correctly classified both words of the multiword phrase.

For the Soham News classification, most errors occurred for similar news categories, predicting national instead of international. International is the minority class, contributing only 4% of the training data. After reviewing the misclassified instances, we identified that these national and international samples mainly fall under the political domain. While the national news articles discuss news and events relevant to the country, international news articles cover news and events outside the country. The only way to distinguish these news categories is by utilizing cue words and contexts, such as person entities referring to national or international political figures, and location entities, such as country names. It becomes difficult for the models to distinguish these categories, especially in articles where national political figures and locations appear in international news articles. Therefore, our models misclassified such international news articles as national ones, resulting in lower accuracy for the international class. Similar problems were observed for minority classes in BBC News classification dataset. Social, China, multimedia, and learning English news categories account for less than 2% of the BBC News training data.

We analyzed predictions of pre-trained models on Discourse analysis and observed most confusing discourse modes are informative and descriptive. Both of our models misclassified 70% informative examples as descriptive ones. Informative sentences provide information to the reader about an entity or a situation. On the other hand, descriptive sentences illustrate specific locations in the story with detailed descriptions to help the reader visualize the scene. These errors might have occurred because these informative examples present detailed information about an entity, thus mis-

understood as descriptive examples. Misclassifications are not likely due to fewer examples of informative class because both models can successfully classify examples in other category, having the least number of training instances.

The COPA task appears to be one of the more challenging tasks where our best-performing models' accuracy (60.80%) is far from human-level performance (99%). Our pre-training data were mainly collected from online news newspaper corpus, often containing crime reports referring to violent incidents. It may impact the reasoning process of the pre-trained language models. The crime news genre typically requires more complicated and sophisticated reasoning, which may not be generalized as day-to-day reasoning. Given two conceivable alternatives and one premise, our models chose the one more relevant to crimes or violent incidents. One misclassified example had the premise "I emptied my pockets." Two provided alternatives were "I retrieved a ticket stub" and "I got a weapon." Our model predicted the incorrect one, "I got a weapon," as the most probable one, whereas the correct causal effect of the premise was "I retrieved a ticket stub." Similarly, the given premise and two choices were "Passenger limit has been reached", "The patrol agent checked his passport", and "Patrol agent accused him of smuggling," respectively. Our model predicted the wrong option, "Patrol agent accused him of smuggling," as the most probable consequence. Moreover, other errors were made due to the incompetency of the model in unambiguously interpreting the premise and choices. For example, one premise stated, 'The office was closed' and provided two choices: 'It was a holiday.' and 'It was summer.' The correct cause was 'It was a holiday,' but our model picked the other wrong option, 'It was summer.' From these observations, the lesson is that the pre-training corpus should adequately cover generalizable common-sense reasoning examples. Such pre-training data might make the model competent to perform day-to-day reasoning and understand textual premises and alternatives unambiguously.

Finally, Bengali cloze-style QA is the most challenging task, which focuses on determining whether language models can serve as a knowledge base. For instance, one

sample article masks an organizational entity and discusses a specific service provided by that entity. The challenging part was three entities (BRAC, Robi, Google) out of four possible options belong to the same entity type - organization (ORG). Only the fourth option was an out-of-context entity (Kathmandu indicating a location (LOC) type entity), which was comparatively easier to filter out while selecting the best alternative for the masked token. Finally, all these articles were collected from Wikipedia, and some of these entities mentioned in the article never appeared in the Bengali pre-training corpus. Therefore, it was challenging for the models to act as a knowledge base, so fine-tuning the language models on Wikipedia articles before adding task-specific layers can further boost performance.

5.4 Bilingual Unigram BERT

Hindi has a higher number of unlabeled and labeled data compared to Bengali. In our pre-training dataset, the Hindi corpus has 1.88M samples more than Bengali. Therefore, we wanted to see whether a model pre-trained on bilingual data could benefit from both languages. We pre-trained the Unigram BERT model on the bilingual corpus obtained by concatenating Bengali and Hindi pre-training data. We used the same hyperparameters that were used for pre-training the Bengali and Hindi Unigram BERT. We increased the vocabulary size to 50,000 to accommodate vocabulary from both languages. We fine-tuned Bilingual Unigram BERT on Bengali and Hindi datasets and reported results in table 5.9.

The results in table 5.9 show that Bilingual Unigram BERT improved performance in Bengali datasets. In fact, it exceeds Bengali and Hindi Unigram BERT by 2.89 points in the Cloze-style QA dataset. This improvement might have resulted from more named entities appearing in the bilingual corpus, which helped the model to predict masked entities in the text correctly. However, this improvement comes with a cost as the performance of the Bilingual Unigram BERT slightly drops in Hindi datasets except for IITP Movie Reviews. In particular, performance drops in COPA

Dataset	Bengali & Hindi Unigram BERT	Bilingual Unigram BERT	Language
Named Entity Recognition	71.54±0.1	**71.85±0.1**	Bengali
Soham News Article	91.29±0.1	**91.64±0.1**	Bengali
Cloze-style QA	56.13±0.1	**59.02±0.1**	Bengali
BBC News Classification	**76.67±0.2**	76.56±0.2	Hindi
IITP Movie Reviews	67.03±0.3	**68.06±0.3**	Hindi
Midas Discourse	**81.42±0.1**	80.74±0.1	Hindi
COPA	**60.84±0.6**	58.35±0.6	Hindi
Average	72.13	**72.32**	-

Table 5.9: Accuracy comparison between Bengali and Hindi Unigram BERT and Bilingual Unigram BERT, except NER task, which compares F1 score.

and Midas discourse datasets but remains comparable for BBC News dataset. Interestingly, Bilingual BERT improves performance by 1 point in the IITP Movie Reviews dataset. Since Hindi has more data availability, this approach can enable knowledge transfer from both languages to improve performance in Bengali datasets with less data availability.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

To address the specific needs of Sanskrit-originated languages, we have presented Bengali and Hindi Unigram and character-level tokenizers and pre-trained models. We show that these tokenizers are well suited for low-resource Bengali and Hindi languages by demonstrating improved performance in a diverse set of downstream tasks. Both tokenizers, especially Bengali and Hindi character-level tokenizer, show robustness over the original BERT tokenizer in highly erroneous and inflected settings. These experimental results illustrate the importance of understanding language-specific needs and choosing a tokenization method accordingly.

The Bengali and Hindi character-level tokenizer can integrate fine-grained intra-word information for constructing word representations. Thus, it might be better equipped to attend compound characters, modified vowels, and consonants. Bengali and Hindi Unigram tokenizer finds the most probable split into tokens to produce more meaningful units and better separate roots, affixes, and modified vowels. Therefore, it might help the NLU model to learn intra-word dependencies better and understand how words are formed in highly inflected and morphologically rich Bengali and Hindi languages.

6.2 Future Work

In future research, we plan to run experiments on more Bengali and Hindi NLU benchmarks and pre-train models on larger document-level corpora with longer sequences. Since Hindi and Bengali have a lot of dialects, we plan to pre-train models on informal Bengali and Hindi texts besides formal texts. It is possible to experiment with the zero-shot cross-lingual transfer because Hindi has comparatively more data availability than Bengali. So, we plan to fine-tune Bilingual Unigram BERT on the Hindi dataset and test on the Bengali equivalent version to see whether the bilingual model can transfer cross-lingual knowledge.

In addition, we intend to use a more efficient pre-training objective, like replaced token detection and span masking instead of tokens. We believe the span masking strategy is better suited for Bengali and Hindi loosely bound word ordering characteristics. Although word order is loosely bound at the local word group level, intra-chunk words always tend to stay together. So, we can mask local word groups as a span, thus enabling the NLU model to learn how local word groups or constituent chunks are formed. Besides span masking, we can modify positional embeddings to handle such relatively free word-ordering characteristics.

Future research can apply similar tokenizers to more recent and larger NLU models (like RoBERTa [16], DeBERTa [61]). We can experiment with Unigram and character-level tokenizers for other languages with similar characteristics, such as Urdu, Assamese, Nepali, Odia, Gujarati, Marathi, and Punjabi. It is also possible to fuse character-level and subword-level information from character-level and Unigram tokenizers and introduce an innovative pre-training strategy to optimize such combined tokenizer-based language models. Finally, we can incorporate similar tokenization strategies for Bengali and Hindi natural language generation (NLG) models.

Chapter 7

Limitations

For the pre-training corpus, we carefully considered data sources with minimum offensive texts. However, there might be unpleasant content like objectionable text and sociocultural or stereotypical biases. Such biases can contribute to biased word representations and have a negative impact, especially for text generation purposes. In addition to acquiring linguistic knowledge such as syntax, structure, and grammar, data-driven pre-trained language models can learn factual or relational knowledge from the pre-trained corpus [62]. This association learning might be helpful for downstream tasks like cloze-style question-answering but can also propagate biases to target tasks. For example, state-of-the-art like GPT-3 demonstrates biases [63] like relating men with jobs requiring higher levels of education or expertise. Such biases can impact downstream NLP tasks. However, recent literature [62] shows that we can change task-specific adaptation methods to mitigate such harmful effects on society in downstream applications. For instance, fine-tuning a smaller set of parameters, like the final layer, and keeping the rest of the parameters fixed can make the model more robust to such biases.

Another limitation of our work is that our pre-training corpus does not include emojis. Therefore, our tokenizers and pre-trained models may not be able to capture additional contextual information encoded in emojis. Even the original BERT vocabulary does not have any textual emoticons present in its vocabulary. Emoji-rich

datasets are found mainly in social media contexts. Emoji plays a vital role in sentiment analysis and conversational settings. For example, facial emojis can indicate positive or negative emotions [64] and are usually common on social media posts. Understanding emojis can further improve the performance of language models in such downstream tasks [65]. One possible solution for this problem is to add new tokens representing emojis to the model's vocabulary and further fine-tune the model on an emoji-featured dataset. The first step would not be required for the Bengali and Hindi Character BERT model as it is not limited to a fixed vocabulary. Hence, the second step of fine-tuning the model on an emoji-rich dataset will suffice.

We used a maximum sequence length of 128 for pre-training our models due to the limitation of computing resources, especially GPU memory constraints. As a result, our model's accuracy might not reach its highest potential for tasks requiring long-range dependencies. However, pre-training the models on longer sequences can improve the performance of document-level tasks. Document-level tasks include news article genre classification and cloze-style QA from Wikipedia articles.

Finally, we compare our pre-trained model's performance with results published in other papers. We noticed that previous papers did not include standard deviation values from different folds or random runs, limiting measures of dispersion.

Bibliography

[1] Wikipedia, *List of languages by number of native speakers — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=List_of_languages_by_number_of_native_speakers&oldid=1175765045, [Online; accessed 9-October-2023].

[2] P. Joshi, S. Santy, A. Budhiraja, K. Bali, and M. Choudhury, "The state and fate of linguistic diversity and inclusion in the NLP world," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 6282–6293. DOI: 10.18653/v1/2020.acl-main.560. [Online]. Available: https://aclanthology.org/2020.acl-main.560.

[3] J. F. Staal, "Sanskrit and sanskritization," *The Journal of Asian Studies*, vol. 22, no. 3, pp. 261–275, 1963.

[4] C. Rahman, M. H. Rahman, M. Rafsan, M. E. Ali, S. Zakir, and R. Muhammod, "CNN for modeling Sanskrit originated Bengali and Hindi language," in *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Online only: Association for Computational Linguistics, Nov. 2022, pp. 47–56. [Online]. Available: https://aclanthology.org/2022.aacl-main.4.

[5] M. Sinha, T. Dasgupta, and A. Basu, "Effect of syntactic features in bangla sentence comprehension," in *Proceedings of the 13th International Conference on Natural Language Processing*, 2016, pp. 275–284.

[6] A. S. Bick, G. Goelman, and R. Frost, "Hebrew brain vs. english brain: Language modulates the way it is processed," *Journal of Cognitive Neuroscience*, vol. 23, no. 9, pp. 2280–2290, 2011.

[7] S. Bhattacharya, M. Choudhury, S. Sarkar, and A. Basu, "Inflectional morphology synthesis for bengali noun, pronoun and verb systems," in *Proc. of the National Conference on Computer Processing of Bangla (NCCPB 05)*, 2005, pp. 34–43.

[8] A. Vaswani *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[9] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392. DOI: 10.18653/v1/D16-1264. [Online]. Available: https://aclanthology.org/D16-1264.

[10] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1112–1122. DOI: 10.18653/v1/N18-1101. [Online]. Available: https://aclanthology.org/N18-1101.

[11] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 353–355. DOI: 10.18653/v1/W18-5446. [Online]. Available: https://aclanthology.org/W18-5446.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. [Online]. Available: https://aclanthology.org/N19-1423.

[13] A. Bhattacharjee *et al.*, "BanglaBERT: Language model pretraining and benchmarks for low-resource language understanding evaluation in Bangla," in *Findings of the Association for Computational Linguistics: NAACL 2022*, Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 1318–1327. DOI: 10.18653/v1/2022.findings-naacl.98. [Online]. Available: https://aclanthology.org/2022.findings-naacl.98.

[14] H. El Boukkouri, O. Ferret, T. Lavergne, H. Noji, P. Zweigenbaum, and J. Tsujii, "CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters," in *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 6903–6915. DOI: 10.18653/v1/2020.coling-main.609. [Online]. Available: https://aclanthology.org/2020.coling-main.609.

[15] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[16] Y. Liu *et al.*, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[17] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*, 2020.

[18] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. [Online]. Available: https://aclanthology.org/D14-1162.

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, 2013. arXiv: 1301.3781 [cs.CL].

[20] M. E. Peters *et al.*, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 2227–2237. DOI: 10.18653/v1/N18-1202. [Online]. Available: https://aclanthology.org/N18-1202.

[21] I. Ashrafi *et al.*, "Banner: A cost-sensitive contextualized model for bangla named entity recognition," *IEEE Access*, vol. 8, pp. 58 206–58 226, 2020. DOI: 10.1109/ACCESS.2020.2982427.

[22] K. I. Islam, S. Kar, M. S. Islam, and M. R. Amin, "SentNoB: A dataset for analysing sentiment on noisy Bangla texts," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 3265–3271. DOI: 10.18653/v1/2021.findings-emnlp.278. [Online]. Available: https://aclanthology.org/2021.findings-emnlp.278.

[23] A. Conneau *et al.*, "Unsupervised cross-lingual representation learning at scale," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 8440–8451. DOI: 10.18653/v1/2020.acl-main.747. [Online]. Available: https://aclanthology.org/2020.acl-main.747.

[24] D. Kakwani *et al.*, "IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, Online: Association for Computational Linguistics, Nov. 2020, pp. 4948–4961. DOI: 10.18653/v1/2020.findings-emnlp.445. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.445.

[25] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[26] Y. Wu *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[27] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, "spaCy: Industrial-strength Natural Language Processing in Python," 2020. DOI: 10.5281/zenodo.1212303.

[28] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 66–75. DOI: 10.18653/v1/P18-1007. [Online]. Available: https://aclanthology.org/P18-1007.

[29] T. Chung and D. Gildea, "Unsupervised tokenization for machine translation," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, 2009, pp. 718–726.

[30] S. Bird and E. Loper, "NLTK: The natural language toolkit," in *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 214–217. [Online]. Available: https://aclanthology.org/P04-3031.

[31] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of English: The Penn Treebank," *Computational Linguistics*, vol. 19, no. 2, J. Hirschberg, Ed., pp. 313–330, 1993. [Online]. Available: https://aclanthology.org/J93-2004.

[32] Wikipedia, *Language change — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=Language_change&oldid=1191987263, [Online; accessed 4-January-2024].

[33] Y. Shibata *et al.*, "Byte pair encoding: A text compression scheme that accelerates pattern matching," 1999.

[34] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 66–71. DOI: 10.18653/v1/D18-2012. [Online]. Available: https://aclanthology.org/D18-2012.

[35] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967. DOI: 10.1109/TIT.1967.1054010.

[36] R. K. Srivastava, K. Greff, and J. Schmidhuber, *Highway networks*, 2015. arXiv: 1505.00387 [cs.LG].

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980.

[38] Y. Zhu *et al.*, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 19–27. DOI: 10.1109/ICCV.2015.11.

[39] C. R. Rahman, M. H. Rahman, M. Rafsan, S. Zakir, M. E. Ali, and R. Muhammod, *CNN for Modeling Sanskrit Originated Bengali and Hindi Language Dataset*, Oct. 2022. DOI: 10.5281/zenodo.7137398. [Online]. Available: https://doi.org/10.5281/zenodo.7137398.

[40] M. Rahman, *Prothom alo bangla news paper*, 2017. [Online]. Available: https://www.prothomalo.com/.

[41] T. I. Khalidi, *Bd news articles*, 2017. [Online]. Available: https://bdnews24.com/.

[42] Gaurav, *Hindi wikipedia articles - 172k*, 2019. [Online]. Available: https://www.kaggle.com/datasets/disisbig/hindi-wikipedia-articles-172k.

[43] A. Thakur, *Hindi oscar corpus*, 2019. [Online]. Available: https://www.kaggle.com/datasets/abhishek/hindi-oscar-corpus.

[44] O. Bojar, V. Diatka, P. Straňák, A. Tamchyna, and D. Zeman, *HindEnCorp 0.5*, LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University, 2014. [Online]. Available: http://hdl.handle.net/11858/00-097C-0000-0023-625F-0.

[45] L. Barrault *et al.*, "Findings of the 2019 conference on machine translation (WMT19)," in *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 1–61. DOI: 10.18653/v1/W19-5301. [Online]. Available: https://aclanthology.org/W19-5301.

[46] A. Mohiuddin, *Daily nayadiganta*, 2019. [Online]. Available: https://www.dailynayadiganta.com/.

[47] S. Shekhar, *Hindi news livehindustan*, 2018. [Online]. Available: https://www.livehindustan.com/.

[48] B. Jain, *Rajashthan patrika epaper:hindi*, 2018. [Online]. Available: https://epaper.patrika.com/.

[49] T. Wolf *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6. [Online]. Available: https://aclanthology.org/2020.emnlp-demos.6.

[50] A. Paszke *et al.*, *Pytorch: An imperative style, high-performance deep learning library*, 2019. arXiv: 1912.01703 [cs.LG].

[51] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[52] M. A. Islam, M. F. Kabir, K. Abdullah-Al-Mamun, and M. N. Huda, "Word/phrase based answer type classification for bengali question answering system," in *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, 2016, pp. 445–448. DOI: 10.1109/ICIEV.2016.7760043.

[53] R. Karim *et al.*, "A step towards information extraction: Named entity recognition in bangla using deep learning," *Journal of Intelligent & Fuzzy Systems*, vol. 37, no. 6, pp. 7401–7413, 2019.

[54] S. Chatterjee, *Classification: Bengali news articles (indicnlp)*, 2019. [Online]. Available: https://www.kaggle.com/datasets/csoham/classification-bengali-news-articles-indicnlp.

[55] *Hasoc: Hindi hate speech dataset*, 2109. [Online]. Available: https://hasocfire.github.io/hasoc/2019/dataset.html.

[56] M. S. Akhtar, A. Kumar, A. Ekbal, and P. Bhattacharyya, "A hybrid deep learning architecture for sentiment analysis," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 482–493. [Online]. Available: https://aclanthology.org/C16-1047.

[57] A. Gordon, Z. Kozareva, and M. Roemmele, "SemEval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning," in **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, Montréal, Canada: Association for Computational Linguistics, 2012, pp. 394–398. [Online]. Available: https://aclanthology.org/S12-1052.

[58] S. Dhanwal *et al.*, "An annotated dataset of discourse modes in Hindi stories," English, in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, Marseille, France: European Language Resources Association, May 2020, pp. 1191–1196, ISBN: 979-10-95546-34-4. [Online]. Available: https://aclanthology.org/2020.lrec-1.149.

[59] M. Roemmele, C. A. Bejan, and A. S. Gordon, "Choice of plausible alternatives: An evaluation of commonsense causal reasoning," in *2011 AAAI Spring Symposium Series*, 2011. [Online]. Available: https://people.ict.usc.edu/~gordon/copa.html.

[60] *Indicglue datasets*, 2020. [Online]. Available: https://ai4bharat.iitm.ac.in/indic-glue.

[61] P. He, X. Liu, J. Gao, and W. Chen, *Deberta: Decoding-enhanced bert with disentangled attention*, 2021. arXiv: 2006.03654 [cs.CL].

[62] F. Ladhak *et al.*, "When do pre-training biases propagate to downstream tasks? a case study in text summarization," in *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, 2023, pp. 3198–3211.

[63] W. Guo and A. Caliskan, "Detecting emergent intersectional biases: Contextualized word embeddings contain a distribution of human-like biases," in *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, ser. AIES '21, New York, NY, USA: Association for Computing Machinery, 2021, 122–133, ISBN: 9781450384735. DOI: 10.1145/3461702.3462536. [Online]. Available: https://doi.org/10.1145/3461702.3462536.

[64] V. A. Pfeifer, E. L. Armstrong, and V. T. Lai, "Do all facial emojis communicate emotion? the impact of facial emojis on perceived sender emotion and text processing," *Computers in Human Behavior*, vol. 126, p. 107 016, 2022, ISSN: 0747-5632. DOI: https://doi.org/10.1016/j.chb.2021.107016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0747563221003393.

[65] P. Delobelle and B. Berendt, *Time to take emoji seriously: They vastly improve casual conversational models*, 2019. arXiv: 1910.13793 [cs.CL].

Appendix A: Fine-tuning hyperparameters and Computation time

A.1 Bengali and Hindi Unigram BERT

Table A.1 shows the training Hyperparameters of the Bengali and Hindi Unigram BERT model. In the following table, we report the learning rate (LR), batch size (BS), max epoch (ME), weight decay (WD), and learning rate warmup ratio (WR). We also provide information about the time required (TR) to run individual experiments, including training time and hyperparameter search (minutes).

A.2 Bengali and Hindi Character BERT

Table A.2 shows the training Hyperparameters of the Bengali and Hindi Character BERT model. In the following table, we report the learning rate (LR), batch size (BS), max epoch (ME), weight decay (WD), and learning rate warmup ratio (WR). We also provide information about the time required (TR) to run individual experiments, including training time and hyperparameter search (minutes).

A.3 Model Quality Analysis and Pre-training Computation time

We also provide information about the time required to run model quality analysis and pre-training experiments (hours) in tables A.3 and A.4.

Task	**Corpus**	**LR**	**BS**	**ME**	**WD**	**WR**	**TR**
Question Classification	Bengali Question Classify	3e-5	32	4	0.01	0.00	30
Named Entity Recognition	Bangla NER dataset	2e-5	16	4	0.01	0.06	87
Genre Classification	Soham News Articles	3e-5	32	6	0.01	0.00	45
	BBC News Articles	3e-5	32	4	0.01	0.05	27
Hate Speech Detection	Hindi Hate Speech dataset	3e-5	32	3	0.01	0.00	48
Sentiment Analysis	Hindi Product Reviews	3e-5	16	4	0.01	0.00	33
	IITP Movie Reviews	3e-5	32	5	0.02	0.05	24
Natural Language Inference (NLI)	COPA dataset	5e-5	32	2	0.01	0.10	21
Discourse Analysis	MIDAS Discourse dataset	3e-5	32	3	0.01	0.10	27
Question Answering	Cloze-style Multiple Choice QA	2e-5	32	4	0.01	0.00	180

Table A.1: Hyperparmaters for fine-tuning Bengali and Hindi Unigram BERT on eight diverse downstream tasks.

Task	**Corpus**	**LR**	**BS**	**ME**	**WD**	**WR**	**TR**
Question Classification	Bengali Question Classify	3e-5	32	6	0.01	0.00	41
Named Entity Recognition	Bangla NER dataset	3e-5	32	3	0.01	0.06	90
Genre Classification	Soham News Articles	3e-5	32	5	0.01	0.00	57
	BBC News Articles	2e-5	32	6	0.01	0.00	33
Hate Speech Detection	Hindi Hate Speech dataset	3e-5	32	3	0.01	0.00	41
Sentiment Analysis	Hindi Product Reviews	3e-5	16	4	0.01	0.00	24
	IITP Movie Reviews	2e-5	32	3	0.01	0.00	24
Natural Language Inference (NLI)	COPA dataset	5e-5	32	2	0.01	0.00	27
Discourse Analysis	MIDAS Discourse dataset	2e-5	32	3	0.01	0.00	27
Question Answering	Cloze-style Multiple Choice QA	2e-5	32	4	0.01	0.00	270

Table A.2: Hyperparmaters for fine-tuning Bengali and Hindi Character BERT on eight diverse downstream tasks.

Corpus	**Unigram BERT**	**Character BERT**	**baseline BERT**
BDNews24	14 hours	19 hours	14 hours
Prothom Alo	24 hours	33 hours	24 hours
Naya Diganta	1 hour	2 hours	1 hour
Livehindustan	8 hours	11 hours	8 hours
Hindi Patrika	1 hour	2 hours	1 hour

Table A.3: Model quality analysis computation time. Naya Diganta and Hindi Patrika were only used for testing purposes.

Pre-training corpus	**Unigram BERT**	**Character BERT**
Bengali corpus	61 hours	73.5 hours
Hindi corpus	73.5 hours	90 hours

Table A.4: Pre-training time for different models.

Appendix B: Examples for Tokenizers

B.1 Wordpiece Tokenizer

B.1.1 Corpus

We employ a toy example with a small corpus to demonstrate how the Wordpiece algorithm works. The toy corpus contains only three sentences with thirty words, given below.

"this is a demonstration of how subword tokenizer works in practice. this is a toy example for explanation purposes. two subword level tokenization methods can be explained using this example."

B.1.2 Vocabulary

For this toy example, we have specified the vocabulary size to be 65 for the Wordpiece tokenizer. In this example, we arbitrarily select 65 as the vocabulary size. It is typically chosen based on downstream validation performance and computation cost. Moreover, we calculate how often each word appears to obtain token frequencies later. The word frequency table B.1 presents words and their counts, respectively.

As explained in section 3.1, the algorithm starts with a small vocabulary consisting of all characters in the corpus. Thus, the words are converted to a sequence of characters, and non-beginning letters of words are preceded by a '##' to indicate that they occur after the first letter. Now, the initial vocabulary is shown below.

Word	**Frequnecies**
this	3
is	2
a	2
demonstration	1
of	1
how	1
subword	2
tokenizer	1
works	1
in	1
practice	1
.	3
toy	1
example	2
for	1
explanation	1
purposes	1
two	1
level	1
tokenization	1
methods	1
can	1
be	1
explained	1
using	1

Table B.1: Word statistics.

[##a, ##b, ##c, ##d, ##e, ##f, ##g, ##h, ##i, ##k, ##l, ##m, ##n, ##o, ##p, ##r, ##s, ##t, ##u, ##v, ##w, ##x, ##y, ##z, ., a, b, c, d, e, f, h, i, l, m, o, p, s, t, u, w]

B.1.3 Merge Rules

This algorithm will expand the initial vocabulary to a pre-defined vocabulary size by learning merge rules to combine pairs of tokens. The initial characters of the base vocabulary are considered tokens. Wordpiece will calculate a score for each pair of tokens from the existing vocabulary using the equation below.

$$Score = \frac{freq(t_1 t_2)}{freq(t_1) \times freq(t_2)} \tag{B.1}$$

We calculate and include scores for ten pairs as examples in table B.2.

pair	**scores**
t##h	0.107
i##s	0.061
d##e	0.077
o##f	1.000
##i##s	0.025
##e##m	0.026
##m##o	0.022
##o##n	0.024
##n##s	0.008
##h##i	0.068

Table B.2: Scores obtained from first ten pairs.

For example, the token pair 't##h' appears three times in the word 'this'. In addition, the individual tokens 't', and '##h' occur 7 and 4 times, respectively. From the frequency table B.1, we can observe that the words 'this' (3), 'tokenizer' (1), 'toy' (1), 'two' (1), and 'tokenization' (1) have the letter 't' in the beginning;

thus, 't' appears seven times in total. On the other hand, '##h' has a prefix '##' indicating it's a non-beginning character that appears four times in the middle of the words 'this' (3) and 'methods' (1). So, we compute the score as follows.

$$Score = \frac{freq(t\#\#h)}{freq(t) \times freq(\#\#h)} = \frac{3}{7 \times 4} = 0.107 \tag{B.2}$$

Once the scores are calculated for all pairs, it will choose the pair with the best score for merging, which is 'o##f' with a score of 1.0. This score can be calculated from the frequency table B.1. 'o' appears at the beginning of the word 'of' one time, and '##f' appears only once in the middle of the word 'of'. The equation emphasizes 'o##f' because 'o' and '##f' as individual tokens are less frequent, occurring once in the corpus. We compute the score as below, and 'of' is added to the vocabulary.

$$Score = \frac{freq(o\#\#f)}{freq(o) \times freq(\#\#f)} = \frac{1}{1 \times 1} = 1.0 \tag{B.3}$$

We continue appending pairs with the highest scores, applying merge rules, until we reach the pre-defined vocabulary size of 65. We save only the final resulting vocabulary but not the merge rules. The final vocabulary is shown here, including all the special tokens for the BERT model.

[[PAD], [UNK], [CLS], [SEP], [MASK], ##a, ##b, ##c, ##d, ##e, ##f, ##g, ##h, ##i, ##k, ##l, ##m, ##n, ##o, ##p, ##r, ##s, ##t, ##u, ##v, ##w, ##x, ##y, ##z, ., a, b, c, d, e, f, h, i, l, m, o, p, s, t, u, w, of, su, sub, pu, subw, ex, ##pl, ##mpl, expl, pr, pur, purp, pra, prac, pract, exa, exampl, expla, ca]

B.1.4 Tokenizing Word

We can tokenize any word once the tokenizer is trained on the toy corpus. We tokenize the word 'examples' using the tokenizer. Given the final vocabulary, the Wordpiece tokenizer will search through all the tokens in the vocabulary and select the longest available token at the beginning of the word. Since the prefix '##' indicates the

token appears in the middle of the word, not at the beginning, the tokenizer does not consider tokens with prefixes like '##e' while searching. Among the possible options 'e', 'ex', 'exa','exampl', it will find and split the longest one 'exampl'. For the remaining split '##es', the tokenizer will again identify the longest possible token, '##e'. Finally, the last remaining split '##s' is already in the vocabulary. Therefore, 'examples' is tokenized as ['exampl', '##e', '##s'].

B.1.5 Tokenizing Out-of-vocabulary Word

The out-of-vocabulary word occurs when the Wordpiece tokenizer encounters subwords not present in the vocabulary. For example, tokenizing the word 'join' will result in [UNK], thus indicating an unknown word. While the tokens '##o', '##i', and '##n' are already present, it cannot find the beginning subword 'j' in the vocabulary.

B.2 Unigram Tokenizer

B.2.1 Vocabulary

To demonstrate the unigram algorithm, we use the same toy corpus. As explained in the section 3.2, we initialize a seed vocabulary with 200 tokens, much larger than our predefined vocabulary size. In this toy example, we arbitrarily chose 65 as the predefined vocabulary size. Vocabulary size is typically chosen based on downstream validation performance and computation cost. We include all 24 basic characters and 176 most common substrings from the corpus in the seed vocabulary. The seed vocabulary is shown here.

[_, t, h, i, s, a, d, e, m, o, n, r, f, w, u, b, k, z, p, c, y, x, l, v, _t, th, wo, or, is, on, ti, wor, _w, _e, _ex, ex, pl, _th, _i, at, ati, atio, ation, tio, tion, io, ion, word, ord, rd, _to, to, le, es, _thi, _this, thi, this, hi, his, _is, _a, ra, ho, _s, _su, _sub, _subw, _subwo, _subwor, _subword, su, sub, subw, subwo, subwor, subword, ub, ubw, ubwo, ubwor, ubword, bw, bwo, bwor, bword, _tok, _toke, _token, _tokeni, _tokeniz, tok, toke, token, tokeni, tokeniz, ok, oke, oken, okeni, okeniz, ke, ken, keni, keniz, en, eni, eniz, ni, niz, iz, _wo, _wor, in, _p, _exa, _exam, _examp, _exampl, _example, exa, exam, examp, exampl, example, xa, xam, xamp, xampl, xample, am, amp, ampl, ample, mp, mpl, mple, ple, _f, _exp, _expl, _expla, exp, expl, expla, xp, xpl, xpla, pla, la, an, se, _wi, _wit, _with, wi, wit, with, it, ith, ds, _d, _de, _dem, _demo, _demon, _demons, _demonst, _demonstr, _demonstra, _demonstrat, _demonstrati, _demonstratio, _demonstration, de, dem, demo, demon, demons, demonst, demonstr, demonstra, demonstrat, demonstrati, demonstratio, demonstration, em, emo, emon, emons, emonst, emonstr, emonstra, emonstrat, emonstrati, emonstratio, emonstration, mo, mon, mons]

Table B.3 shows some example tokens present in the seed vocabulary and their

frequencies. The first letter of each word starts with a '_' because each space was replaced with a '_' during the pre-tokenization step. Token frequencies are obtained from the word frequencies in the corpus (see the table B.4 below).

Token	**Frequnecies**
_t	7
th	6
wo	5
or	5
is	4
on	4
ti	4
wor	4
_w	4
_e	4
_ex	4
ex	4
pl	4
_th	3
_i	3
at	3
_tokeniz	2
atio	3
ation	3
tio	3

Table B.3: Token statistics.

Word	**Frequnecies**
this	3
is	2
a	2
demonstration	1
of	1
how	1
subword	2
tokenizer	1
works	1
in	1
practice	1
toy	1
example	2
for	1
explanation	1
purposes	1
two	1
level	1
tokenization	1
methods	1
can	1
be	1
explained	1
using	1

Table B.4: Word statistics.

B.2.2 Calculating Loss

We need to compute corpus-level loss for the current vocabulary at each training step. To calculate loss, we compute probabilities for each token in the vocabulary. Similar to the unigram language model, each token is considered independent, so we can estimate the probability for each token by dividing the token frequency by the sum of all token frequencies in the corpus. We report a log of probabilities to avoid numerical underflow. Moreover, negative log values are recorded so that negative log values can be used directly to estimate the negative log-likelihood during loss calculation. The first 20 probability values are reported in the table B.5.

For example, the token '_tokeniz' occurs a total of 2 times in the corpus in the words 'tokenizer' (1) and 'tokenization' (1), respectively. Then, the token frequency can be divided by the sum of all token frequencies (568) in the corpus to obtain token probability, shown below.

$$P(\_tokeniz) = -log\left(\frac{freq(\_tokeniz)}{freq(sum)}\right) = -log\left(\frac{2}{568}\right) = 5.649 \tag{B.4}$$

Similarly, we can calculate the probability for token 'ation', which occurs three times in the words 'demonstration' (1), 'explanation' (1), and 'tokenization' (1), respectively. The calculation is shown below.

$$P(ation) = -log\left(\frac{freq(ation)}{freq(sum)}\right) = -log\left(\frac{3}{568}\right) = 5.244 \tag{B.5}$$

Once we are done estimating token probabilities, we have all the required values to calculate loss. Each word in the corpus is tokenized using available vocabulary for calculating loss. Given token probabilities, the Viterbi algorithm is applied to consider the most likely segmentation of each word into sub-word tokens. For example, if we tokenize the word 'tokenization' with the current vocabulary, we will get the most probable splits ['_tokeniz', 'ation']. As tokens are considered independent, we can multiply token probabilities to obtain word probability, which is equivalent to addition in log space. So, the negative log probability for the word 'tokenization' will

Token	Probabilities
t	3.634
h	4.396
i	3.703
s	3.703
a	3.944
d	4.550
e	3.346
m	4.956
o	3.509
n	4.040
r	4.145
f	5.244
w	4.145
u	5.244
b	5.244
k	5.244
z	5.649
p	4.396
_tokeniz	5.649
ation	5.244

Table B.5: Token probabilities.

be the sum of the log probabilities of '_tokeniz' and 'ation', which is 10.893. Like 'tokenization', we apply a similar process to get negative log probabilities for each word. The table B.6 shows each word's negative log probabilities and frequencies. Finally, we multiply this word level log probability values with corresponding word frequencies from the table to obtain final scores. The corpus level loss is the sum of

Word	**Frequencies**	**Word Probabilites**
_this	2	5.649
_is	2	5.649
_a	2	5.649
_demonstration	1	6.342
_of	1	11.598
_how	1	12.639
_subword	2	5.649
_tokenizer	1	13.140
_works	1	14.596
_in	1	8.495
_practice	1	30.087
_toy	1	11.586
_example	1	5.649
_for	1	10.382
_explanation	1	14.932
_purposes	1	31.889
_with	2	5.649
_few	1	13.140
_words	1	11.298
_two	1	9.129
_level	1	22.328
_tokenization	1	10.892
_methods	1	24.856
_can	1	13.738
_be	1	11.436
_explained	1	19.195
_these	1	13.833
_examples	1	9.352

Table B.6: Word probabilities.

these final scores computed using the equation below.

$$loss = \sum_{i=1}^{n} freq_i * (-log(P(w_i))) \tag{B.6}$$

We get the following if we sum all the values in the table B.6.

$$loss = freq(\_this) * (-log(P(\_this)) + \cdots + freq(\_examples) * (-log(P(\_examples)) \tag{B.7}$$

$$loss = 2 \times 5.649 + \cdots + 1 \times 9.352 = 387.022 \tag{B.8}$$

B.2.3 Discarding Tokens

After determining corpus-level loss, we need to determine how much loss increases for removing each token from the vocabulary. To obtain individual token-level loss, we need to estimate loss after removing that token from the vocabulary. Then, we can subtract the corpus-level loss from the loss without that particular token, indicating the increase in loss for that token. In the table B.7, we have included token-level loss for some example tokens.

Instead of removing one token at a time, we discard 10% of tokens having the least impact on corpus-level loss at each training step. This shrinking factor, which represents the percentage of tokens to discard, is chosen based on the final corpus-level loss. 10% yields the best corpus-level loss for this corpus. In the first iteration, 20 tokens were removed, as shown in the table B.8. Table B.7 shows that the token 'ation' increases the loss by 7.889 because words like 'explanation', and 'tokenization' cannot be tokenized with the best probable split without this token. Given the probability values, the best segmentation for token 'tokenization' and 'explanation' are ['_tokeniz', 'ation'] and ['_expla', 'n', 'ation']. Thus, the algorithm preserves tokens necessary for tokenization purposes.

From the table B.8, we can observe the token '_ex' has a zero loss and can be removed from the vocabulary. This token appears at the beginning of the word 'examples'. It can be discarded because the best possible segmentation of 'examples' is

Token	**Loss**
_t	2.083
th	1.224
wo	2.921
or	2.921
_ex	0.000
on	0.000
ti	2.381
wor	0.000
_w	0.000
_e	0.000
mo	0.000
mon	0.000
mons	0.000
_tokeniz	5.691
th	1.224
_t	2.083
_th	2.152
ho	2.256
ation	7.889
mo	0.000

Table B.7: Increase in loss for removal of each token.

['_example', 's'], which does not include the token '_ex'. We continue discarding 10% of tokens with minimum impact on loss in each iteration until we reach the desired vocabulary size of 65. The final vocabulary is shown below, where five spots are preserved for special tokens. For tokenization purposes, we save both the vocabulary and scores.

Token	Loss
is	0.0
on	0.0
wor	0.0
_w	0.0
_e	0.0
_ex	0.0
ex	0.0
pl	0.0
_i	0.0
at	0.0
ati	0.0
atio	0.0
tio	0.0
tion	0.0
io	0.0
ion	0.0
word	0.0
ord	0.0
rd	0.0
to	0.0

Table B.8: Tokens with least impact on corpus-level loss.

[[PAD], [UNK], [CLS], [SEP], [MASK], _, t, h, i, s, a, d, e, m, o, n, r, f, w, u, b, k, z, p, c, y, x, l, v, _t, th, wo, or, ti, _th, ation, _to, le, es, _this, _is, _a, ra, ho, _subword, _tokeniz, _wor, in, _p, _example, _f, _expla, an, _with, ds, _demonstration, emonstr, emonstra, emonstrat, emonstrati, emonstratio, emonstration, mo, mon, mons]

B.2.4 Tokenizing Word

Given the vocabulary and scores, we can tokenize any word. If we tokenize the word 'examples', the tokenizer will split it into subword tokens ['_example', 's']. Therefore, the unigram tokenizer can better split the root '_example' and the suffix 's'. However, the Wordpiece tokenizer fails to do that by tokenizing the word 'examples' as ['exampl', '##e', '##s']. Similarly, the unigram tokenizer will also tokenize 'subwords' into most likely split ['_subword', 's'], thus splitting into root and suffix. On the contrary, Wordpiece tokenizer tokenizes 'subwords' as ['subw', '##o', '##r', '##d', '##s']

B.2.5 Tokenizing Out-of-vocabulary Word

The unigram tokenizer will tokenize the word 'join' into [UNK] despite tokens 'o', 'i', and 'n' being present in the vocabulary, as the beginning subword '_j' is absent.