Pronunciation-Aware Syllable Tokenizer for Nepali Automatic Speech Recognition System

Rupak Raj Ghimire[1], Bal Krishna Bal[1], Balaram Prasain[2] and Prakash Poudyal[1]

[1] Information and Language Processing Research Lab (ILPRL)
Department of Computer Science and Engineering
Kathmandu University, Nepal
[2] Central Department of Linguistics
Tribhuvan University, Nepal
rughimire@gmail.com, bal@ku.edu.np, balaram.prasain@cdl.tu.edu.np, prakash@ku.edu.np

Abstract

The Automatic Speech Recognition (ASR) has come up with significant advancements over the course of several decades, transitioning from a rule-based method to a statistical approach, and ultimately to the use of end-to-end (E2E) frameworks. This phenomenon continues with the progression of machine learning and deep learning methodologies. The E2E approach for ASR has demonstrated predominant success in the case of resourceful languages with larger annotated corpus. However, the accuracy is quite low for low-resourced languages such as Nepali. In this regard, language-specific tools such as tokenizers seem to play a vital role in improving the performance of the E2E model for low-resourced languages like Nepali.

In this paper, we propose a pronunciation-aware syllable tokenizer for the Nepali language which improves the results of the E2E model. Our experiment confirm that the introduction of the proposed tokenizer yields better performance with the Character Error Rate (CER) 8.09% compared to other language-independent tokenizers.

Keywords: Syllable Tokenizer, Nepali ASR, GRU, RNN, E2E

1 Introduction

Automatic Speech Recognition (ASR) systems have been developed using acoustic, language, and lexical models, employing techniques such as as Hidden Markov Mode (HMM), Gaussian Mixture Mode (GMM), statistical, and probabilistic models traditionally. In deep learning based models, Recurrent Neural Network (RNN) is a popular choice for ASR to capture temporal dependencies of speech data. One of the major advantages of RNNs for ASR is that they can be trained end-to-end (E2E), meaning that the entire system is trained from raw audio input to generate text as output without the need of hand-engineered features. In End-to-End (E2E) ASR, the neural network directly learns to map the input audio signal to the corresponding text sequence without the need for intermediate representations. This simplifies the ASR development process by eliminating the need for hand-crafted features and complex models including Acoustic Model, Language Model, Lexicon etc. The Gated Recurrent Uni (GRU)(Cho et al., 2014) and Long Short-Term Memory (LSTM)(Hochreiter and Schmidhuber, 1997) are the variations of RNN. As per Chung et al. (2014) GRU is more effective in terms of computation whereas the LSTM is better suited for larger datasets.

The transcribed text is considered as a label for supervised learning and needs accurate alignment with speech data. In the context of RNN-based classification, the transcribed text is required to convert into the tokens and them into numerical labels. The tokens are smaller constituent segments of the text. However, the substantial disparity in abstraction levels between the audio signal provided as input and the linguistic tokens generated as output poses a significant challenge for a E2E model to acquire the necessary representations. The common way to tokenize is via the use of language-independent tokenizer such as character-based, Byte Pair Encodin (BPE) tokenizers. The use of the various tokenization techniques in ASR are explored by researchers Higuchi et al. (2022) and Singh et al. (2021). The character tokenizer is the simplest tokenizer. The BPE (Gage, 1994) segments the word into the subwords based on the co-occurrence of the characters and used as tokenization(Sennrich et al., 2016) scheme. SentencePiece (Kudo

and Richardson, 2018) is another language-independent subword tokenizer that can be trained on the raw sentences.

Beside the language independent tokenizers, a pronunciation - aware syllable tokenizer is used to divide words into smaller units (phonetic units) called syllables. This type of tokenizer is capable of breaking the text into the language-specific phonetic unit which ensures the appropriate phonetic alignment of token with speech signal. This tokenizer can improve the accuracy of ASR and Text-to-Speech (TTS) systems by more accurately aligning the segmentation with natural speech patterns and making it easy to classify them using E2E models.

The rest of the paper is organized as follows the related works and the dataset used are explained in section 2 and 3 respectively. Section 4 explains the working of the pronunciation-aware syllable tokenizer and ASR model. Section 5 presents the conducted experiments and their results. Finally, the paper concludes with section 6 where summary of findings, future plans, and potential extensions to the work are explained.

2 Related works

There are various ways of tokenizing the transcribed text. The language independent tokenizers such as character tokenizer, word, and subword tokenizers (BPE, Unigram) are commonly used to reduce the complexity of the tokenization. The character based tokenizer is simplest form of the tokenization scheme on Natural Language Processing (NLP) task.

The use of different text tokenizers for ASR has been studied for various languages. Higuchi et al. (2022) has experimented with subword unit-based acoustically aligned tokenization techniques. Recently, Shen et al. (2023) has explored the use of pronunciation-aware unique character encoding techniques for RNN-based ASR models. The experiment showed that the pronunciation-aware character encoding perform better. For low-resource languages, Diwan and Jyothi (2021) experimented with the reduce and reconstruct approach to minimize the number of prediction errors. The number of features was decreased by replacing them with characters that sound similar. Once prediction is made the replaced characters were substituted to their original form.

Kudo and Richardson (2018) also studied a different tokenizer called SentencePiece tokenizer and discovered that this type of tokenizer can be helpful for E2E models. Papadourakis et al. (2021) also conducted experiments on Phonetically Induced Subwords for the ASR system and observed that Connectionist Temporal Classification (CTC) and RNN-T architectures showed a performance increase of 15.21% compared to other methods.

Kanda et al. (2016) has described the syllable-based system for the Japanese language and implemented it as a subword language model (SLM) to coordinate the CTC based Acoustic Model (AM) score. Using a syllable-based system on the CTC decoding phase improved the performance of the system. Similarly Zou et al. (2018) also explored the use of various tokenizers such as character, syllable, and context-dependent phoneme(CDP) in Mandarin speech recognition. This research concluded with a better result when a syllable-based unit is applied on the CTC based ASR system. Lightweight Wordpiece Model is proposed by Xu et al. (2021), which takes word and all training vocabulary as an input and return the longest sub-word segmentation of the input word. Patel et al. (2020) used phonetic alignment technique for the machine transliteration task. As per the authors' experiment the use of the phonetic alignment is superior over phrase-based alignment approach and result is improved around 50% on Indic languages. Anoop and Ramakrishnan (2023) studied the modeling of unit for E2E speech recognition task and concluded that the syllable-based units are best choice in context of the Indian Languages.

The most research that has been done on the Nepali ASR system development have used character based tokenizer (Paudel et al., 2023; Joshi et al., 2023; Bhatta et al., 2020;Regmi and Bal, 2021;Dhakal et al., 2022). Regmi et al. (2019) has used one-gram text tokenizer on proposed RNN and CTC-based ASR model. The authors prepared own dataset by extracting text containing 1320 words and recording with three speakers. They achieved Character Error Rate (CER) of 34%. Similarly,

Bhatta et al. (2020) and Regmi and Bal (2021) used the character based tokenizer on Convolution Neural Network (CNN), GRU, and CTC-based ASR model and CTC-Attention-based Encoder-Decoder model respectively. Their reported CER is 11% and 10.3% respectively. Dhakal et al. (2022) also used character based tokenizer on CNN, Bi-LSTM and ResNet based and found CER of 17.06%. The recent experiment reported by Paudel et al. (2023) has used CNN-Transformer based E2E model and achieved 11.14% CER.

Research on the other possible tokenizers for Nepali script is not well explored. The experiment conducted on various tokenization techniques (Patel et al., 2020;Anoop and Ramakrishnan, 2023;Si et al., 2023;Shen et al., 2023;Xu et al., 2019) clearly demonstrate that the tokenizer plays vital role in the performance of the ASR and that has to be explored on the Nepali language as well. Beside that, our another motivation for developing the tokenizer is based on the fact that phonetically rich languages like Nepali, the conventional tokenizer (word, subword, BPE, Unigram or character based) generated tokens do not properly align with the acoustic feature. The proposed syllabic tokenizer split the given text into the syllable which are phonetically aware. The tokenizer can be used for the the language that uses the Devanagari script.

3 Dataset

The Open SLR dataset (Kjartansson et al., 2018) is used[1] in this work for comparative study of the proposed and other tokenizer on E2E Nepali ASR System (explained in section 4.3). This dataset contains raw recordings in *flac* encoding and transcribed text in the Devanagari script. The metadata of the speech corpus are summarized in Table 1.

The Open SLR dataset for Nepali speech has three major problems- 1) background noise 2) large gaps in between the word speech, and 3) not normalized transcribed text. For noise reduction we generated a time-smoothed spectrogram using an adaptive Infinite Impulse Response (Adaptive-IIR) digital filter (Kwan, 2001) and mask is computed on this spectrogram, then the mask is smoothed with a filter over frequency and time. This mask is applied to the spectrogram of the signal. Finally, the signal is recovered. Using this approach the white noise is reduced. We trimmed the start and end of the signal to remove starting and ending silent parts of the speech.

[1]http://openslr.com/54

Table 1: Open SLR (Kjartansson et al., 2018) speech corpus summary

Particular	**Details**
Unique Utterance	86062
Total Utterance	∼157K
Duration of Recording	∼9200 minutes
Total Number of Speakers	527

The clean dataset is available in our research lab of the University (the link is anonymize to meet the requirement of the blind review).

4 Methodology

4.1 Tokenization

Tokenization can be done in various ways. The simple and widely used tokenization technique for ASR is a character-based tokenizer. This tokenizer is not suitable for those languages which have complex and conjunct characters and combination of the characters is considered as single syllable. The Nepali language follows the Devanagari script which has vowels, consonants, vowel markers, and special characters as shown in the Table 2. The vowel markers in Devanagari script play a crucial role in accurately representing and adding vowel sounds to the consonant characters, helping to distinguish between words that have the same consonant sounds but different vowel sounds. Beside the regular characters (listed in Table 2) there are other complex characters such as क्ष, त्र, ज्ञ, त्त, द्ध, श्र, द्य which are combination of the consonant, vowel and special markers (eg. क्ष = क + ् + ष).

Let us consider the word क्रान्ति, if we use character based segmentation we end up with tokens {क, ्, र, ा , ... }. The tokens ् and ा do not have any phonetic value unless they are combined with respective consonants. When we use other tokenizer such as BPE on the Devanagari script we end up with tokens such as {क्र , ान्ति}. As per the word morphology the vowel marker ा should not appear

Table 2: Nepali script characters

Type	Symbols
Vowel	अ, आ, ई, इ, उ, ऊ, ए, ऐ, ऋ, ओ, औ, ऋ, ऌ
Consonants	क, ख, ग, घ, ङ, च, छ, ज,झ, ञ, ट, ठ, ड, ढ, ण, त, थ, द, ध,न, प, फ, ब, भ, म, य, र, ल, व, श, ष, स, ह
Vowel markers	ा, ि, ी, ु, ू, े, ै, ो, ौ
Other markers	् , ।, ं , ृ etc.
Numbers	०, १, २, ३, ४, ५, ६, ७, ८, ९

before consonant. In this example, tokens {क्रा , न् , ति} are more phonetic then other alternative. So, it is clear that if we need to tokenize to the syllabic level the existing tokenizers will not work.

To address these types of problems we need a more specialized tokenizer. We have proposed the sliding window-based tokenization algorithm which relies on the pre-calculated Syllable Dataset. For syllables dataset preparation we used grammar rules of the Devanagari script (Acharya, 1974). The syllable dataset is used as a lookup for the tokenizer. The algorithm is presented in Algorithm 1 and the working mechanism is shown in Figure 1. From the Figure 1 we can observe that given the input sentence क्षेत्रमा कसैका, the expected output of the tokenizer is क्षे त्र मा क सै का. The abstract view of the algorithm proposed in Algorithm 1 looks similar to other (Xu et al., 2021) syllabic tokenization approach. But the fundamental different is we use separate syllable dataset for the lookup purpose. The choice of the sub-word or syllable is based on those dataset rather than a longest sub-word as suggested by Xu et al. (2021). The syllable dataset itself covers the phonetic essences of Nepali language.

Choosing the window size: The Window size (*win_size*) is important parameter for Algorithm 1. When *win_size* is less than the characters involved in composition of complex syllable the tokenizer end up with inappropriate syllable and sometimes the algorithm end up with error. So, we took the help of a linguist expert to appropriate the output of the tokenizer.

Algorithm 1: Syllabic Tokenizer for Nepali

Input: Sentence : S
Output: Tokens : FT
Data:
Syllable Dataset : SD
Window Size: win_size

```
FT = [ ]
T = [ordered list of all characters in S]
while current_win_pos > len(T) do
    t_window = T[current_win_pos : (current_win_pos + win_size)]
    ct = { all possible syllables from t_window each starting 0^th position }
    foreach ct_cur in ct do
        if ct_cur in SD then
            current_win_pos += len(ct_cur)
            FT.append(ct_cur)
            break
return FT
```

4.2 Working of Syllabic Tokenizer

First of all, the given sentence $S = (C_1C_2C_3...C_n)$ is tokenized using the character-based tokenizer to form the tokens T which is an ordered list of all characters C_n as shown in equation (1). A maximum of 4 characters are used to form the complex syllables so the window size is considered as 4. The window sliding starts from the first token. On the first window, the first four tokens from T are captured within the window i.e. (C_1, C_2, C_3, C_4). For those tokens within the window W_1 we can generate the combined tokens CT_{w1} preserving the order of the tokens as shown in equation (2).

Then we can check the existence of the each item of CT_{w1} on syllables dataset (SD) starting from index 0. If we find 1^{st} item on the lookup then we add this to the list of a final token $FT = \{C_1C_2C_3\}$ and move to the second window CT_{w2} which starts from $N = 3$ as we already considered three items of list T - this can be represented as in equation (3).

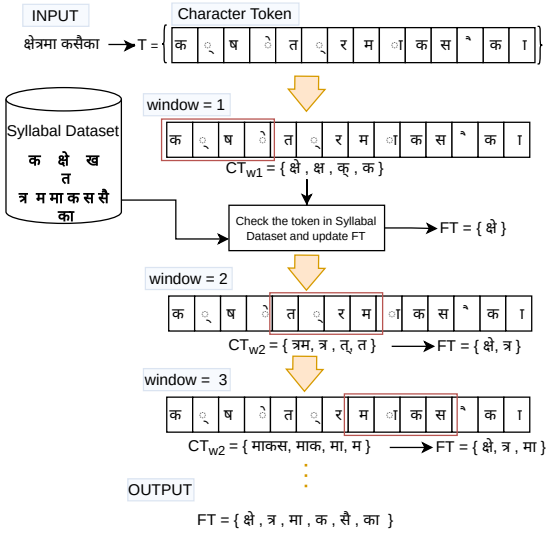

Figure 1: Working of pronunciation-aware syllable tokenize for the Nepali language

$$T = list(C_1, C_2 ... C_N) \quad (1)$$

$$CT_{w1} = \{C_1C_2C_3C_4, C_1C_2C_3, C_1C_2, C_1\} \quad (2)$$

$$CT_{w2} = \{C_4C_5C_6C_7, C_4C_5C_6, C_4C_5, C_4\} \quad (3)$$

For next iteration, if the valid syllable is $C_4C_5C_6C_7$ then we update final token as $FT = \{C_1C_2C_3, C_4C_5C_6C_7\}$ and mark the next position to start the windows as we covered seventh character within the valid token. This way we can iterate over all the positions to find the valid syllables out of the available tokens T.

In some window segment (say w_x) there is a chance to get more than one valid token that is present in the lookup. For example - S = प्रियतम, CT_{w_x} = {'प्रि' , 'प्र' , 'प्' , 'प'} . Here 'प्रि' , 'प्र' , 'प' are valid syllable in Nepali but 'प्रि' will be chosen which is present in the SD. As the order of the characters is preserved while generating the tokens (Equation 2) and lookup is done from 1^{st} position, our tokenizer choose 'प्रि' on the first iteration and move to the next window. Some of the outputs of the tokenizer are listed in Table 3.

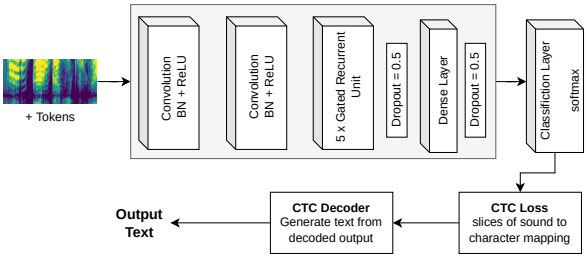

Figure 2: E2E ASR Model used for the experiment

Table 3: tokenizer outputs

Input	Generated tokens
व्यक्तित्वमा प्रभाव पर्ने	['व्', 'य', 'क्', 'ति', 'त्', 'व', 'मा', ' ', 'प्र', 'भा', 'व', ' ', 'प', 'र्', 'ने']
घाउ लागेको क्षेत्रमा	['घा', 'उ', ' ', 'ला', 'गे', 'को', ' ', 'क्षे', 'त्र', 'मा']

4.3 Baseline ASR Model

In order to measure the effectiveness of proposed tokenizer we developed the E2E ASR model for Nepali using Tensorflow (Abadi et al., 2015). We adopted CNN based E2E ASR model proposed by Amodei et al. (2015) which is also similar with the model used in (Regmi et al., 2019), (Bhatta et al., 2020), and (Regmi and Bal, 2021) so that we compare the end results. The Figure 2 shows the architecture of the E2E model. The model accepts the spectrogram and tokens of transcribed text as input pair and passes them to the two convolution layer with following parameters:

$$\begin{pmatrix} filters = 32 \\ kernelsize = [11, 41] \\ strides = [2, 2] \end{pmatrix} \begin{pmatrix} filters = 32 \\ kernelsize = [11, 21] \\ strides = [1, 2] \end{pmatrix}$$

Batch normalization (BN) is done after each convolution layer to speed up the training by normalizing the raw data. Rectified Linear Unit (ReLU) is used after normalization. Then the RNN is designed using a five (5) GRU layer with *tanh* as an activation function and *sigmoid* as recurrent activation. The dropout factor of 0.5 is used. The *softmax* is applied to the classification layer. CTC decoder is used for generating the text from the classifier output.

5 Experiment and Results

The E2E ASR model as explained in section 4.3 and depicted in Figure 2 is trained with

publicly available speech corpus from Open SLR (Kjartansson et al., 2018). The tokens are produced by tokenizers such as BPE, Character, Unigram-based tokenizers and our proposed tokenizer. This gave us a solid base for evaluating the proposed tokenizer. For BPE and Unigram tokenization we used SentencePiece (Kudo and Richardson, 2018) library. The token size is summarize in Table 4.

Table 4: Token Size

Tokenization Method	#Tokens
a) Unigram tokenizer	997
b) BPE tokenizer	1000
c) Character tokenizer	60
d) Pronunciation-aware syllable tokenizer	650

The E2E ASR model as depicted in Figure 2 has been developed using the Tensorflow (Abadi et al., 2015). The model training and running of the experiment has been done on the machine with RTX 3090 GPU (24GB GPU Memory), Ryzen 9 5600x CPU and 32 GB memory. The Open SLR dataset (Kjartansson et al., 2018) for Nepali language does not have standard separated training and validation data set. So the training and validation dataset is made using a 9:1 split ratio (train : validation).The training and validation loss of all the training has been plotted in Figure 3.

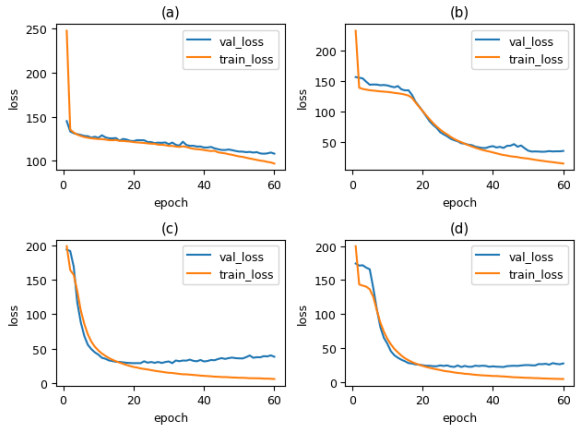

Figure 3: Training vs. Validation loss during training using: a) Unigram tokenizer, b) BPE tokenizer c) Character tokenizer, and d) Pronunciation-aware syllable tokenizer

For the evaluation, the character error rate (CER) and word error rate (WER) are used (Morris et al., 2004). CER is used to evaluate the performance of automatic speech recognition (ASR) systems by assessing how well they convert speech input into accurate textual output. A lower CER indicates higher accuracy and better performance. On the other hand, WER considers individual words instead of characters.

Table 5: Experiment Result

Tokenization Method	WER	CER
a) Unigram tokenizer	91.1%	83.5%
b) BPE tokenizer	53.3%	18.1%
c) Character tokenizer	40.1%	9.3%
d) Pronunciation-aware syllable tokenizer	**36.33%**	**8.09%**

Table 5 shows the comparison of CER and WER on the model trained with the proposed tokenizer and other tokenizers. Our experiment resulted in increased accuracy of **8.09%** CER and **36.33%** WER on the proposed pronunciation-aware syllable tokenizer. The closest CER and WER are that of the character-based tokenizer, respectively, (9.3%, 40.1%) which is 1.21% and 3.77% higer than the CER and WER scores of our tokenizer. The WER is higher because we have not applied any language model at this stage for correcting the output.

If we compare CER with other independent research such as the one conducted by Regmi and Bal (2021), the performance has improved by **2.21%**.

6 Conclusion and Future Work

We conducted an experiment using a syllable tokenizer that incorporates pronunciation awareness to segment Nepali text into subword units or syllables. We applied this tokenizer to an E2E ASR framework for aligning transcribed text with speech signals. After introducing the tokenizer in the pipeline, we found that the performance of the model outperformed the state-of-the-art character-based tokenizer by 1.21% and 3.77% in terms of CER and WER respectively. This improvement proves that the syllable-based tokenizer which is pronunciation aware is very crucial for phonetically rich and morphologically complex

languages like Nepali.

There are several areas for improvement in this research. We have plans to use this tokenizer to further investigate its role in various E2E-based models and frameworks like AED, T-AED, and RNN-T. Similarly, hyperparameter-based estimation can be performed on the existing model to determine the best values for parameters such as kernel size and number of epochs needed. The Open SLR dataset (Kjartansson et al., 2018) for Nepali language does not have standard separated training and validation data set. In the next phase, we are planning to augment more data and investigate this dataset and propose the appropriate training and test split.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, and et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.

Shivaraja Acharya. 1974. *Nepali Barnecharan Shikshya, 2031(BS)*. Sajha Prakasana, Nepal.

Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, and et al. 2015. Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. *CoRR*, abs/1512.02595.

Chandran Savithri Anoop and Angarai Ganesan Ramakrishnan. 2023. Suitability of syllable-based modeling units for end-to-end speech recognition in Sanskrit and other Indian languages. *Expert Systems with Applications*, 220:119722.

Bharat Bhatta, Basanta Joshi, and Ram Krishna Maharjhan. 2020. Nepali speech recognition using cnn, gru and ctc. In *Taiwan Conference on Computational Linguistics and Speech Processing*.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In *NIPS 2014 Workshop on Deep Learning*.

Manish Dhakal, Arman Chhetri, Aman Kumar Gupta, Prabin Lamichhane, Suraj Pandey, and Subarna Shakya. 2022. Automatic speech recognition for the Nepali language using CNN, bidirectional LSTM and ResNet. In *2022 International Conference on Inventive Computation Technologies (ICICT)*, pages 515–521.

Anuj Diwan and Preethi Jyothi. 2021. Reduce and Reconstruct: ASR for Low-Resource Phonetic Languages. In *INTERSPEECH 2021*.

Philip Gage. 1994. A New Algorithm for Data Compression. *The C Users Journal*, 12:23–38.

Yosuke Higuchi, Keita Karube, Tetsuji Ogawa, and Tetsunori Kobayashi. 2022. Hierarchical Conditional End-to-End ASR with CTC and Multi-Granular Subword Units. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7797–7801.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural Computation*, 9:1735–80.

Basanta Joshi, Bharat Bhatta, and Ram Krishna Maharjan. 2023. End to End based Nepali Speech Recognition System. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 17(102).

Naoyuki Kanda, Xugang lu, and Hisashi Kawai. 2016. Maximum a posteriori Based Decoding for CTC Acoustic Models. In *INTERSPEECH 2016*, pages 1868–1872.

Oddur Kjartansson, Supheakmungkol Sarin, Knot Pipatsrisawat, Martin Jansche, and Linne Ha. 2018. Crowd-Sourced Speech Corpora for Javanese, Sundanese, Sinhala, Nepali, and Bangladeshi Bengali. In *Proc. The 6th Intl. Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU)*, pages 52–55, Gurugram, India.

Taku Kudo and John Richardson. 2018. 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*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

H.K. Kwan. 2001. Adaptive iir digital filters for noise and echo reduction in speech. In *2001 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (IEEE Cat. No.01CH37233)*, volume 1, pages 47–50 vol.1.

Andrew Cameron Morris, Viktoria Maier, and Phil D. Green. 2004. From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition. In *INTERSPEECH 2004*. ISCA.

Vasileios Papadourakis, Markus Müller, Jing Liu, Athanasios Mouchtaris, and Maurizio Omologo. 2021. Phonetically Induced Subwords for End-to-End Speech Recognition. In *INTERSPEECH 2021*, pages 1992–1996.

Parth Patel, Manthan Mehta, Pushpak Bhattacharya, and Arjun Atreya. 2020. Leveraging Alignment and Phonology for low-resource Indic to English Neural Machine Transliteration. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, pages 373–378, Indian Institute of Technology Patna, Patna, India. NLP Association of India (NLPAI).

Shishir Paudel, Bal Krishna Bal, and Dhiraj Shrestha. 2023. Large Vocabulary Continous Speech Recognition for Nepali Language using CNN and Transformer. In *Proceedings of the 4th Conference on Language, Data and Knowledge*, pages 328–333, Vienna, Austria. NOVA CLUNL, Portugal.

Paribesh Regmi, Arjun Dahal, and Basanta Joshi. 2019. Nepali Speech Recognition using RNN-CTC Model. *International Journal of Computer Applications*, 178(31):1–6.

Sunil Regmi and Bal Krishna Bal. 2021. An End-to-End Speech Recognition for the Nepali Language. In *Proceedings of the 18th International Conference on Natural Language Processing (ICON)*, pages 180–185, National Institute of Technology Silchar, Silchar, India. NLP Association of India (NLPAI).

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Peng Shen, Xugang Lu, and Hisashi Kawai. 2023. Pronunciation-Aware Unique Character Encoding for RNN Transducer-Based Mandarin Speech Recognition. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pages 123–129.

Chenglei Si, Zhengyan Zhang, Yingfa Chen, Fanchao Qi, Xiaozhi Wang, Zhiyuan Liu, Yasheng Wang, Qun Liu, and Maosong Sun. 2023. Sub-Character Tokenization for Chinese Pretrained Language Models. *Transactions of the Association for Computational Linguistics*, 11:469–487.

Sachin Singh, Ashutosh Gupta, Aman Maghan, Dhananjaya Gowda, Shatrughan Singh, and Chanwoo Kim. 2021. Comparative study of different tokenization strategies for streaming end-to-end asr. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 388–394.

Hainan Xu, Kartik Audhkhasi, Yinghui Huang, Jesse Emond, and Bhuvana Ramabhadran. 2021. Regularizing word segmentation by creating misspellings. In *Interspeech 2021*, pages 2561–2565. ISCA.

Hainan Xu, Shuoyang Ding, and Shinji Watanabe. 2019. Improving End-to-end Speech Recognition with Pronunciation-assisted Sub-word Modeling. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7110–7114.

Wei Zou, Dongwei Jiang, Shuaijiang Zhao, and Xiangang Li. 2018. A comparable study of modeling units for end-to-end Mandarin speech recognition. In *11th International Symposium on Chinese Spoken Language Processing (ISCSLP)*.