INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Retraining-free Customized ASR for Enharmonic Words Based on a Named-Entity-Aware Model and Phoneme Similarity Estimation

Yui Sudo[1], *Kazuya Hata*[1], *Kazuhiro Nakadai*[2]

[1]Honda Research Institute Japan Co., Ltd., Saitama, Japan
[2]Department of Systems and Control Engineering, School of Engineering, Tokyo Institute of Technology, Tokyo, Japan

{yui.sudo, kazuya.hata}@jp.honda-ri.com, nakadai@ra.sc.e.titech.ac.jp

Abstract

End-to-end automatic speech recognition (E2E-ASR) has the potential to improve performance, but a specific issue that needs to be addressed is the difficulty it has in handling enharmonic words: named entities (NEs) with the same pronunciation and part of speech that are spelled differently. This often occurs with Japanese personal names that have the same pronunciation but different Kanji characters. Since such NE words tend to be important keywords, ASR easily loses user trust if it misrecognizes them. To solve these problems, this paper proposes a novel retraining-free customized method for E2E-ASRs based on a named-entity-aware E2E-ASR model and phoneme similarity estimation. Experimental results show that the proposed method improves the target NE character error rate by 35.7% on average relative to the conventional E2E-ASR model when selecting personal names as a target NE.

Index Terms: speech recognition, named-entity-aware, phoneme similarity, enharmonic word

1. Introduction

End-to-end automatic speech recognition (E2E-ASR) has been intensively studied [1–6]. This approach combines an acoustic model (AM) and a language model (LM) into a single neural network-based model to improve ASR performance. However, in practical applications of the E2E-ASR system, a specific challenge arises regarding low customizability. As such a problem, we focus on handling words with the same pronunciation and part of speech but are spelled differently, which we call "enharmonic words." The enharmonic words commonly appear in Japanese person names that have the same pronunciation but are represented using different Kanji characters, such as "Abe" (a Japanese person name), which can be multiply represented as follows:

Abe:{阿部, 安部, 安邊, 綾部, etc.}

Personal names are an important named entity (NE), so misrecognition of such words offends users and reduces their trust in the ASR system. In addition, there may be multiple persons in the same group or section with names that read the same but have different Kanji characters. In such cases, the misrecognition of the names can confuse people. These problems are not limited to Japanese, but also exist in other languages more or less in writing of personal names. Since two enharmonic words have the same part of speech (*e.g.* person's name), they are more difficult to handle compared to two homonyms, which may have different parts of speech and can be solved in context. Moreover, because enharmonic words may not be included in the training data, in which case this enharmonic words problem typically includes both in-vocabulary (IV) and out-of-vocabulary (OOV) NE words.

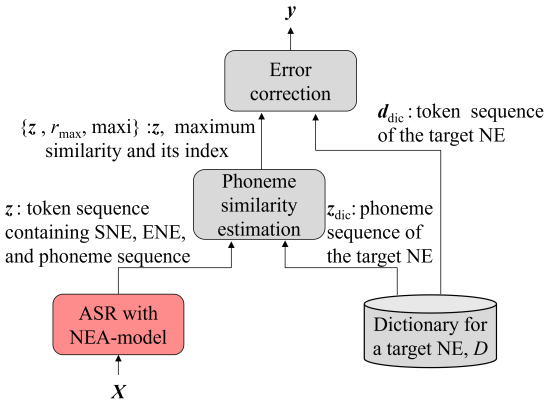


Figure 1: *Overall architecture of the proposed method which consists of ASR with a NEA ASR model, a dictionary, a phoneme similarity estimation, and an error correction.*

A number of methods based on an weighted finite state transducer (WFST) have been proposed to address the customizability issues for non-E2E-ASR [7–11]. The WFST has been used to address pure OOV problems, but it is difficult to address enharmonic word problems [7–9]. The class N-gram [10, 11] allows users to register NE words without huge computational cost such as retraining neural networks, which makes it easier to handle enharmonic words whether they are IV or OOV, because the phoneme sequences can be obtained separately from the AM. However, E2E-ASR is preferable due to its high performance. Methods combining E2E-ASR with WFST have been reported. For example, [12–14] used DNN-WFST models to detect NE words and construct in-class LMs for NE words. Although these methods enable the customization of the target NE words using in-class LMs, it is unable to handle enharmonic words. This is because both phonemes and tokens are necessary to deal with such words while they output only a sequence of tokens rather than that of phonemes.

Several E2E-ASR-based methods without WFST for the customization have been proposed. These can be divided into two categories: retraining-based or retraining-free. The first E2E-based approach involves retraining the E2E-ASR model using text data or an LM. One such method is LM fusion [15, 16]. This method attempts to improve accuracy by combining the E2E-ASR model with another LM and by rescoring the N-best hypotheses generated by the E2E-ASR. However, the effect of LMs connected to the AM in a cascade way is limited, because the original E2E-ASR model is not retrained, and thus the hypotheses to be generated are not updated. Domain adaptation, which has been investigated to overcome this problem, involves directly retraining the ASR model using only text data [17–19]. Compared to LM fusion, errors are less likely to oc-

10.21437/Interspeech.2023-1320

cur, since the E2E-ASR model is directly updated, while the retraining is time-consuming. In addition to these two methods, ASR model adaptation using speech synthesis (TTS) has been reported [20–22]. This TTS-based adaptation uses only text data, but it requires a TTS model. Although these retraining-based methods can achieve the customization, they require certain amount of retraining data and considerable retraining time.

The second approach for the E2E-based customization is retraining-free. Knowledge-based modeling [23] and tree-based contextualization [24] efficiently represent relationships between words through knowledge graphs and tree structures, enabling rare word recognition. Biasing [25–27] allows users to register any phrases in an editable phrase list to bias the ASR model towards the registered phrases. Another approach is the contextual adapter [28], which enables users to register arbitrary words in a catalog list without retraining using a training adapter coupled to a pre-trained E2E-ASR with a small amount of speech data. Although these methods offer flexible customization, they cannot handle pairs of the token and phoneme sequences, which is necessary to handle enharmonic words.

Therefore, this paper proposes a novel customized E2E-ASR model that can handle enharmonic words without the need for retraining. This method is designed to be accessible to users without expertise in ASR or linguistics. The proposed method leverages a named-entity-aware (NEA) ASR model to extract target named entities (NEs) or proper nouns. It then estimates the phoneme similarity between the extracted NE and each word entry in a dictionary containing enharmonic words. If the similarity is high enough, the extracted phoneme sequence is replaced with the dictionary word that has the highest similarity, allowing for improved recognition of the target NE.

2. Preliminary

This section briefly introduces the attention-based Conformer [29, 30], which is used with the proposed method in the ASR system. The Conformer encoder consists of two convolutional layers, a linear projection layer, and a positional encoding layer, followed by Conformer blocks. The Conformer blocks transform an audio feature sequence, $\boldsymbol{X}$, into a hidden state $\boldsymbol{H}$ as,

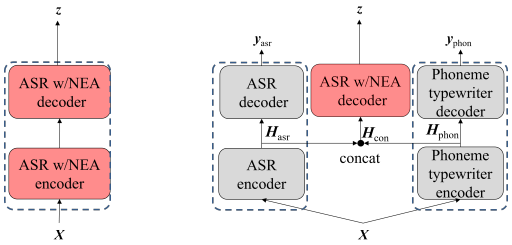
$$\boldsymbol{H} = \mathrm{Encoder}(\boldsymbol{X}). \tag{1}$$

Each Conformer block has a multiheaded self-attention layer, a convolution layer, two linear layers, and a layer-normalization layer [31], with residual connections [32]. Given $\boldsymbol{H}$ generated by the encoder in Eq. (1) and a previously estimated token sequence $\boldsymbol{y}_{s-1} = \{y_1, \cdots, y_{s-1}\}$, the decoder estimates the next token y_s where s represents an output label index. This process is recursively performed as follows:

$$y_s = \mathrm{Decoder}(\boldsymbol{y}_{s-1}, \boldsymbol{H}). \tag{2}$$

The previously estimated token sequence $\boldsymbol{y}_{s-1}$ is first converted into token embeddings. These are then fed into decoder layers with hidden states $\boldsymbol{H}$, followed by a linear projection. The predicted probability of y_s is obtained using a softmax function, given outputs of the linear projection. The decoder layer comprises a self-attention network and source-target attention, followed by a position-wise feed-forward network. The likelihood of the total token sequence $\boldsymbol{y}_S$ consisting of S tokens estimated by an attention model is described as follows:

$$P(\boldsymbol{y}_S \mid \boldsymbol{X}) = \prod_{s=1}^{S} P\left(y_s \mid \boldsymbol{y}_{s-1}, \boldsymbol{H}\right). \tag{3}$$



(a) *Single-encoder NEA.* (b) *Dual-encoder NEA.*

Figure 2: *The network architectures of the NEA models.*

3. Proposed method

The overall architecture of the proposed method is shown in Figure 1. It consists of ASR with a NEA model trained on a corpus tagged with target NEs or proper nouns: a dictionary for target NEs containing enharmonic words, phoneme similarity estimation, and error correction. From the input signal $\boldsymbol{X}$, instead of $\boldsymbol{y}_S$, the ASR with NEA model first estimates an NE-aware token sequence $\boldsymbol{z}$ defined as follows:

$$\begin{aligned} \boldsymbol{z} \quad = \quad & \{y_1, \cdots, y_{s_1-1}, \{\text{`}<\text{'}, y_{s_1}, \cdots, y_{e_1}, \boldsymbol{z}_{\mathrm{p}_1}, \text{`}>\text{'}\}, \\ & y_{e_1+1}, \cdots, y_{s_n-1}, \{\text{`}<\text{'}, y_{s_n}, \cdots, y_{e_n}, \boldsymbol{z}_{\mathrm{p}_n}, \text{`}>\text{'}\}, \\ & y_{e_n+1}, \cdots, y_S\}. \end{aligned} \tag{4}$$

where '<' and '>' represent special tokens that denote the start of NE (SNE) and end of NE (ENE), and y_{s_n} and y_{e_n} $(n = 1, \cdots, N)$ are the start and end tokens for the n-th NE token sequence $\boldsymbol{y}_{\mathrm{NE}_n}$ in $\boldsymbol{z}$. $\boldsymbol{z}_{\mathrm{p}_n}$ is the phoneme sequence corresponding to $\boldsymbol{y}_{\mathrm{NE}_n}$. For example, given the ground truth "my name is $y_1y_2y_3$," we simply insert a special token representing the start and end of the NE word and the corresponding phoneme sequence, as in "my name is $< y_1y_2y_3, z_1z_2z_3 >$." Since the part-of-speech and pronunciation are provided by the dataset, the NEA model can be easily trained. The phoneme similarity estimation then calculates the similarity in the phoneme sequence between the extracted phoneme sequence $\boldsymbol{z}_{\mathrm{p}_n}$ and each word entry in the dictionary, D, defined as follows:

$$D \quad = \quad \{\{\boldsymbol{d}_{\mathrm{NE}_i}, \boldsymbol{z}_{\mathrm{d}_i}\} | i = 1, \cdots, I\}, \tag{5}$$

where $\boldsymbol{d}_{\mathrm{NE}_i}$ and $\boldsymbol{z}_{\mathrm{d}_i}$ represent the i-th NE token sequence and phoneme sequence in D, respectively.

Since the method comprises simple pairs of NE tokens and phoneme sequences, users without expertise in linguistics can easily edit entries. When considering that maxi is the index of D with the highest similarity, the error correction replaces the extracted phoneme sequence, $\boldsymbol{z}_{\mathrm{p}_n}$, with the dictionary word with the highest similarity $\boldsymbol{d}_{\mathrm{NE}_{\mathrm{maxi}}}$ to correct for ASR errors caused by enharmonic words, when the similarity is higher than the threshold, V_{th}. To correctly estimate the phoneme sequences of NEs, we propose two network models: single-encoder NEA (S-NEA) and dual-encoder NEA (D-NEA).

3.1. Single-encoder NEA

The S-NEA model with a single-encoder–decoder network is shown in Figure 2a. This is the same as the one used in conventional ASRs, as described in Section 2. The only difference is that it estimates $\boldsymbol{z}$ instead of $\boldsymbol{y}_S$ as follows:

$$P(\boldsymbol{z} \mid \boldsymbol{X}) = \prod_{s=1}^{|\boldsymbol{z}|} P\left(z_s \mid \boldsymbol{z}_{s-1}, \boldsymbol{H}\right). \tag{6}$$

Note that since the phoneme sequences are added only to NE words using the special tokens SNE and ENE in the training data, the number of phoneme sequences in the training data may not be sufficient. This may reduce the robustness of NE words that contain rarely appearing phonemes in the training data.

3.2. Dual-encoder NEA

The D-NEA consists of two pre-trained encoder-decoder sub-networks for a conventional ASR, and a phoneme typewriter (PT), and an NEA decoder sub-networks, as shown in Figure 2b. The PT is trained to estimate phoneme sequences instead of token sequences [33, 34]. By integrating these sub-networks, the D-NEA is expected to recognize a word even when the word includes phonemes that rarely appear during training. The two encoder-decoder sub-networks have the same network architecture as the S-NEA. Conventional ASR sub-networks estimate token sequences without phonemes, $\boldsymbol{y}_{\rm asr}$, while the PT sub-network estimates phoneme sequences without tokens, $\boldsymbol{y}_{\rm p}$. The outputs of the decoder parts for the conventional ASR and PT, $\boldsymbol{H}_{\rm asr}$ and $\boldsymbol{H}_{\rm p}$, are concatenated into $\boldsymbol{H}_{\rm con} = {\rm concat}(\boldsymbol{H}_{\rm asr}, \boldsymbol{H}_{\rm p})$. The NEA decoder takes $\boldsymbol{H}_{\rm con}$ as an input, and estimates $\boldsymbol{z}$. The model was trained in two stages. First, the ASR and PT sub-networks were trained, and then the NEA decoder was trained while freezing the trained ASR and PT sub-networks. Note that $\boldsymbol{y}_{\rm asr}$ and $\boldsymbol{y}_{\rm p}$ are used to pre-train the conventional ASR and the PT sub-networks, and they are discarded in the later processing, such as decoding.

3.3. Phoneme similarity estimation and error correction

Algorithm 1 shows the phoneme similarity estimation. From the output of ASR with the NEA model, $\boldsymbol{z}$, the n-th phoneme sequence, $\boldsymbol{z}_{{\rm p}_n}$ is extracted based on the special tokens, SNE and ENE (line 2 in Algorithm 1). Then, phoneme similarity, r, between $\boldsymbol{z}_{{\rm p}_n}$ and each entry in the dictionary, $\boldsymbol{z}_{{\rm d}_i} (i = 1, \cdots, I)$, is calculated (lines 3-5 in Algorithm 1). The phoneme similarity, r, is calculated using Gestalt pattern matching [35] described as follows:

$$r = {\rm Sim}(\boldsymbol{z}_{{\rm d}_i}, \boldsymbol{z}_{{\rm p}_n}) = \frac{2K}{|\boldsymbol{z}_{{\rm d}_i}| + |\boldsymbol{z}_{{\rm p}_n}|} \ (0 \leq r \leq 1), \quad (7)$$

where K denotes the number of the matched phonemes. A value of $r = 1$ means that the two phoneme sequences are perfectly matched, while a value of $r = 0$ means that the two phoneme sequences are perfectly different. Then, the NE with the highest phoneme similarity, $r_{\rm max}$, and its index maxi, in the dictionary are selected (line 6 in Algorithm 1). Additionally, a threshold, V_{th}, is introduced to improve the robustness. If the phoneme similarity is greater than V_{th}, the corresponding NE, $y_{{\rm NE}_n}$, is replaced with that of the dictionary, $d_{{\rm NE}_{\rm maxi}}$, and the SNE, ENE, and phoneme sequences are deleted (line 8 in Algorithm 1). Otherwise, the NE, $y_{{\rm NE}_n}$, is not replaced, and the special token and phoneme sequence are deleted (line 10 in Algorithm 1). This threshold prevents the proposed model from replacing an NE word with a non-NE word.

Algorithm 1 Phoneme similarity estimation

1: **for** $n = 1, \cdots N$ **do**
2: $\quad \boldsymbol{z}_{{\rm p}_n} = {\rm ExtractPhoneme}(\boldsymbol{z}, n)$
3: $\quad$ **for** $i = 1, \cdots, I$ **do**
4: $\quad\quad r[i] = {\rm Sim}(\boldsymbol{z}_{{\rm d}_i}, \boldsymbol{z}_{{\rm p}_n})$
5: $\quad$ **end for**
6: $\quad [r_{\rm max}, {\rm maxi}] = \max(r)$
7: $\quad$ **if** $r_{\rm max} > V_{th}$ **then**
8: $\quad\quad \boldsymbol{z} = {\rm ReplaceNEword\text{-}and\text{-}DeletePhoneme}(\boldsymbol{z}, \boldsymbol{d}_{{\rm NE}_{\rm maxi}}, n)$
9: $\quad$ **else**
10: $\quad\quad \boldsymbol{z} = {\rm DeletePhoneme}(\boldsymbol{z}, n)$
11: $\quad$ **end if**
12: **end for**
13: **return** $\boldsymbol{z}$

4. Experiments

We evaluated the proposed S-NEA and D-NEA models in terms of performance, scalability, and robustness of the proposed method. We then analyzed the recognition results for personal names to confirm the effectiveness of the proposed method for enharmonic words. Furthermore, the number of NE words, I, registered in the dictionary and the effect of the threshold, V_{th}, on performance were examined to verify the scalability and robustness of the proposed method.

4.1. Experimental setup

The input feature was an 80-dimensional Mel-scale filter-bank feature with a window size of 512 samples and a hop length of 128 samples. The sampling frequency was 16 kHz. SpecAugment [36] was then used. All encoders in the S-NEA and D-NEA models had the same network structure. Each encoder comprised two convolutional layers with stride two, a 512-dimensional linear projection layer, and a positional encoding layer, followed by 12 Conformer layers with 2,048 linear units and layer normalization. The decoder had six transformer layers with 2,048 units. Both the decoders in the S-NEA and D-NEA models had the same network structures. The dimension size of the attention was 512 with 8-multihead attentions. The proposed model was trained on 50 epochs using the Adam optimizer at a learning rate of 0.0015 with 15,000 warm-up steps.

We used a customized dataset for training. The training dataset consisted of the Corpus of Spontaneous Japanese [37], a multiple speaker speech database developed by the Advanced Telecommunications Research Institute International (ATR-APP) [38], and our in-house dataset. Since the CSJ does not provide "person name" tags, we used MeCab [39], a morphological analysis tool, to tag the training data. Our in-house dataset consists of 93 hours of single-speaker speech data collected from three different locations, including multiple scenarios, such as meetings and morning assemblies.[1] We used two evaluation datasets, our in-house evaluation dataset and CSJ eval3. Personal names were present in 5.8% of the total 4,838 utterances in our in-house evaluation dataset. CSJ eval1 and eval2 were excluded because they contained too few personal names. The character error rate (CER) for the token sequences of the target NEs (CER-NE) and all character sequences (CER-all) were calculated to evaluate the effect of the proposed method. The CER-NE was calculated within a subset of the target NEs, defined as follows:

$$\text{CER-NE} = \frac{S_{\rm NE} + I_{\rm NE} + D_{\rm NE}}{N_{\rm NE}}, \quad (8)$$

where $S_{\rm NE}$, $I_{\rm NE}$, $D_{\rm NE}$, and $N_{\rm NE}$ denote the number of substitutions, insertions, deletions, and the total number of tokens in person names, respectively. The threshold, V_{th}, described in Section 3.3 was set to 0.8. We used the ESPnet [40] toolkit.

[1] Our in-house dataset is not released for confidentiality and privacy reasons.

Table 1: *Main results comparing the CER-all and CER-NE for the in-house dataset and CSJ eval3.*

Method	In-house		CSJ eval3		Average	
	All	NE	All	NE	All	NE
Baseline	6.9	33.5	3.6	38.7	5.4	35.0
S-NEA	**6.8**	23.9	**3.4**	33.6	**5.1**	26.6
D-NEA	7.0	**22.2**	3.5	**23.4**	5.3	**22.5**

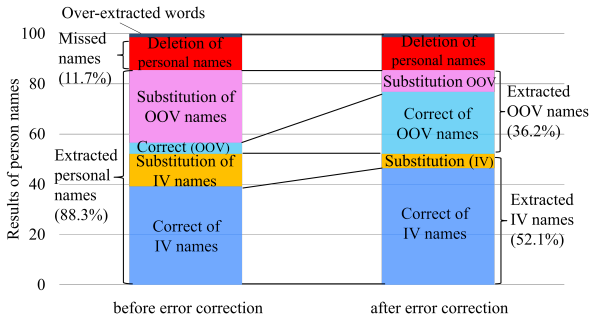


Figure 3: *Breakdown of the results of the target named entities.*

4.2. Comparison between S-NEA and D-NEA

The result of the comparison between the S-NEA and D-NEA models is shown in Table 1. For the two evaluation sets, both of the proposed models outperformed the baseline. In particular, the CER-NE for the D-NEA was 35.7% lower than the baseline. Furthermore, the CER-all was also decreased due to the improved CER-NE. The CER-all for the S-NEA in the CSJ eval3 was 3.4%, which outperformed the state of the art [41]. Comparing the two proposed methods, the D-NEA outperformed the S-NEA for the CER-NE for both evaluation datasets. Although the CER-all of the D-NEA were slightly worse than those of the S-NEA, the improved CER-NE is expected to enhance usability because personal names are extremely important for specific communities and tasks, as described in the introduction.

We analyzed the effectiveness of the proposed method in detail. Figure 3 shows the results for the in-house evaluation dataset using the D-NEA model before and after the error correction. 88.3% of the personal names in the evaluation set were correctly extracted. Of these, 52.1% were IV personal names included in the training data, and the remaining 36.2% were OOV personal names. The substitution errors of the IV personal names were 13.0% before the error correction, which was caused by the enharmonic word problem, and it reduced to be half after the correction using the dictionary, as shown in orange in Figure 3. Most OOV personal names resulted in substitution errors before the error correction, whereas they were well recognized after the correction with the dictionary, as shown in pink in Figure 3.

4.3. Influence of the dictionary size

We tested the effect of the number of personal names, I, as described in Eq. (5) in the dictionary for the CER-NE, since I tends to be large unless the user intentionally erases names that have been registered in the dictionary in the past. Figure 4 shows the impact of I on the CER-NE for the in-house evaluation dataset using the D-NEA model. Since the proposed method is intended to be used in a specific community, a subset of one location was used. When the dictionary was not used (I = 0), the CER-NE was 46.5%. As the number of personal names in the dictionary increased, the CER-NE improved, because the number of enharmonic personal names (IV and OOV) decreased. The CER-NE was the lowest when the dictionary contained the exact same personal names (I = 33) as those used in the evaluation dataset because the number of enharmonic personal names was zero. As I became even larger, the CER-NE gradually increased, but the CER-NE was still better than the case without the dictionary, even when registering 1,000 personal names (I = 1,000) in the dictionary. This shows the robustness of the proposed method against the increased number of registered personal names. The analysis of the data obtained from the users of the proposed system showed that only 5.7% of the utterances contained the personal names. Therefore, the computational cost did not change for the remaining 94.3% of the utterances. The additional computational cost for the utterances containing personal names was only 0.4% of the baseline for a dictionary of 1,000 personal names.

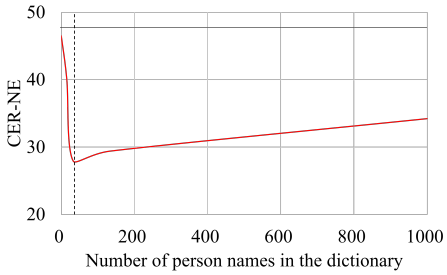


Figure 4: *Effect of the number of words in the dictionary.*

Table 2: *Effect of the threshold.*

Threshold V_{th}	CER-all	CER-NE
0.0	**7.0**	27.6
0.5	**7.0**	**19.8**
0.8	**7.0**	22.2
1.0	7.1	28.6

4.4. Effect of the threshold

The effect of the phoneme similarity threshold, V_{th}, described in Section 3.3, was examined. Table 2 shows the effect of the phoneme similarity threshold, V_{th}, on the CER-NE for the in-house evaluation dataset using D-NEA. When V_{th} = 0.5, the CER-NE was the smallest. When V_{th} = 0.0, all extracted personal names were replaced with the personal names from the dictionary, which caused an error when the NEA model misextracted a non-personal name as a personal name. Conversely, when V_{th} = 1.0, personal names are replaced only when the extracted phoneme sequences exactly matched the phoneme sequence in the dictionary, so even a small error in the phoneme sequence output by the NEA model resulted in the degradation of the CER-NE. Therefore, the threshold, V_{th} should be set by considering the balance between these two errors.

5. Conclusion

This paper presented a retraining-free customizable E2E-ASR model consisting of ASR with an NEA model trained on a corpus tagged with target NEs, a dictionary for a target NE containing enharmonic words, phoneme similarity estimations, and error corrections. The proposed method successfully improved recognition of both IV and OOV enharmonic personal names. We plan to extend the proposed method so that it can be customized for not only personal names but also other NEs.

6. References

[1] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006, pp. 369–376.

[2] A. Graves, "Sequence transduction with recurrent neural networks," in *Proc. ICML*, 2012.

[3] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. ICML*, 2014, pp. 1764–1772.

[4] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," *Proc. NeurIPS*, vol. 28, pp. 577–585, 2015.

[5] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. ICASSP*, 2016, pp. 4960–4964.

[6] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid ctc/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[7] I. Bazzi and J. R. Glass, "Modeling out-of-vocabulary words for robust speech recognition," in *Proc. 6th International Conference on Spoken Language Processing (ICSLP 2000)*, 2000, pp. vol. 1, 401–404.

[8] A. Yazgan and M. Saraçlar, "Hybrid language models for out of vocabulary word detection in large vocabulary conversational speech recognition," in *Proc. ICASSP*, vol. 1, 2004, pp. I–745.

[9] X. Zhang, D. Povey, and S. Khudanpur, "Oov recovery with efficient 2nd pass decoding and open-vocabulary word-level rnnlm rescoring for hybrid asr," in *Proc. ICASSP*, 2020, pp. 6334–6338.

[10] P. F. Brown, V. J. D. Pietra, P. V. de Souza, J. C. Lai, and R. L. Mercer, "Class-based n-gram models of natural language," *Comput. Linguistics*, vol. 18, pp. 467–479, 1992.

[11] A. Horndasch, C. Kaufhold, and E. Nöth, "How to add word classes to the kaldi speech recognition toolkit," in *International Conference on Text, Speech and Dialogue*, 2016.

[12] R. Huang, O. Abdel-Hamid, X. Li, and G. Evermann, "Class lm and word mapping for contextual biasing in end-to-end asr," in *Proc. Interspeech*, 2020, pp. 4348–4351.

[13] I. Williams, A. Kannan, P. Aleksic, D. Rybach, and T. Sainath, "Contextual speech recognition in end-to-end neural network systems using beam search," in *Proc. Interspeech*, 2018.

[14] K. Atsushi, "A study of biasing technical terms in medical speech recognition using weighted finite-state transducer," *Journal of the Acoustical Society of Japan*, vol. 43, pp. 66–68, 2022.

[15] A. Kannan, Y. Wu, P. Nguyen, T. N. Sainath, Z. Chen, and R. Prabhavalkar, "An analysis of incorporating an external language model into a sequence-to-sequence model," in *Proc. ICASSP*, 2018, pp. 5824–5828.

[16] A. Sriram, H. Jun, S. Satheesh, and A. Coates, "Cold fusion: training seq2seq models together with language models," in *Proc. Interspeech*, 2018, pp. 387–391.

[17] J. Pylkkönen, A. Ukkonen, J. Kilpikoski, S. Tamminen, and H. Heikinheimo, "Fast text-only domain adaptation of rnn-transducer prediction network," in *Proc. Interspeech*, 2021, pp. 1882–1886.

[18] X. Chen, Z. Meng, S. Parthasarathy, and J. Li, "Factorized neural transducer for efficient language model adaptation," in *Proc. ICASSP*, 2022, pp. 8132–8136.

[19] H. Sato, T. Komori, T. Mishima, Y. Kawai, T. Mochizuki, S. Sato, and T. Ogawa, "Text-Only Domain Adaptation Based on Intermediate CTC," in *Proc. Interspeech*, 2022, pp. 2208–2212.

[20] C. S. Khe, B. Françoise, B. Arnaud, G. Dhruv, K. Andreas, K. Nikhil, L. Tamar, Z. Petr, Z. Harry, T. J. Leif, M. Giovanni, and Z. Lillian, "Personalization of end-to-end speech recognition on mobile devices for named entities," in *Proc. ASRU*, 2019, pp. 23–30.

[21] X. Zheng, Y. Liu, D. Gunceler, and D. Willett, "Using synthetic audio to improve the recognition of out-of-vocabulary words in end-to-end asr systems," in *Proc. ICASSP*, 2021, pp. 5674–5678.

[22] Y. Deng, R. Zhao, Z. Meng, X. Chen, B. Liu, J. Li, Y. Gong, and L. He, "Improving rnn-t for domain scaling using semi-supervised training with neural tts," in *Proc. Interspeech*, 2021, pp. 751–755.

[23] N. Das, M. Sunkara, D. Bekal, D. H. Chau, S. Bodapati, and K. Kirchhoff, "Listen, know and spell: Knowledge-infused subword modeling for improving asr performance of out-of-vocabulary (oov) named entities," in *Proc. ICASSP*, 2022.

[24] G. Sun, C. Zhang, and P. C. Woodland, "Tree-constrained pointer generator for end-to-end contextual speech recognition," in *Proc. ASRU*. IEEE, 2021, pp. 780–787.

[25] G. Pundak, T. N. Sainath, R. Prabhavalkar, A. Kannan, and D. Zhao, "Deep context: End-to-end contextual speech recognition," in *Proc. SLT*, 2018, pp. 418–425.

[26] M. Jain, G. Keren, J. Mahadeokar, and Y. Saraf, "Contextual rnn-t for open domain asr," in *Proc. Interspeech*, 2020, pp. 11–15.

[27] A. Bruguier, R. Prabhavalkar, G. Pundak, and T. N. Sainath, "Phoebe: Pronunciation-aware contextualization for end-to-end speech recognition," in *Proc. ICASSP*. IEEE, 2019, pp. 6171–6175.

[28] K. M. Sathyendra, T. Muniyappa, F.-J. Chang, J. Liu, J. Su, G. Strimel, A. Mouchtaris, and S. Kunzmann, "Contextual adapters for personalized speech recognition in neural transducers," in *Proc. ICASSP*, 2022.

[29] A. Gulati, C. Chiu, J. Qin, J. Yu, N. Parmar, R. Pang, S. Wang, W. Han, Y. Wu, Y. Zhang, and Z. Zhang, "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, 2020, pp. 5036–5040.

[30] P. Guo, F. Boyer, X. Chang, T. Hayashi, Y. Higuchi, H. Inaguma, N. Kamo, C. Li, D. Garcia-Romero, J. Shi *et al.*, "Recent developments on espnet toolkit boosted by conformer," in *Proc. ICASSP*, 2021, pp. 5874–5878.

[31] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *Proc. NeurIPS*, 2016.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.

[33] W. Chan and I. Lane, "On Online Attention-Based Speech Recognition and Joint Mandarin Character-Pinyin Training," in *Proc. Interspeech*, 2016, pp. 3404–3408.

[34] S. Toshniwal, H. Tang, L. Lu, and K. Livescu, "Multitask learning with low-level auxiliary tasks for encoder-decoder based speech recognition," in *Proc. Interspeech*, 2017, pp. 3532–3536.

[35] J. W. Ratcliff and D. E. Metzener, "Pattern-matching-the gestalt approach," *DDJ*, vol. 13, no. 7, p. 46, 1988.

[36] D. S. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Proc. Interspeech*, 2019, pp. 2613–2617.

[37] K. Maekawa, "Corpus of spontaneous Japanese: Its design and evaluation," in *ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*, 2003.

[38] A. Kurematsu, K. Takeda, Y. Sagisaka, S. Katagiri, H. Kuwabara, and K. Shikano, "Atr japanese speech database as a tool of speech recognition and synthesis," *Speech Communication*, vol. 9, no. 4, pp. 357–363, 1990.

[39] T. Kudo, K. Yamamoto, and Y. Matsumoto, "Applying conditional random fields to Japanese morphological analysis," in *Proc. EMNLP*, 2004, pp. 230–237.

[40] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "ESPnet: End-to-end speech processing toolkit," in *Proc. Interspeech*, 2018, pp. 2207–2211.

[41] S. Karita, Y. Kubo, M. Bacchiani, and L. Jones, "A comparative study on neural architectures and training methods for japanese speech recognition," in *Proc. Interspeech*, 2021, pp. 2092–2096.