Spelling-Aware Word-Based End-to-End ASR

Ekaterina Egorova, Hari Krishna Vydana, Lukáš Burget, and Jan Honza Černocký, *Senior Member, IEEE*

***Abstract*—We propose a new end-to-end architecture for automatic speech recognition that expands the "listen, attend and spell" (LAS) paradigm. While the main word-predicting network is trained to predict words, the secondary, speller network, is optimized to predict word spellings from inner representations of the main network (e.g. word embeddings or context vectors from the attention module). We show that this joint training improves the word error rate of a word-based system and enables solving additional tasks, such as out-of-vocabulary word detection and recovery. The tests are conducted on LibriSpeech dataset consisting of 1000 h of read speech.**

***Index Terms*—ASR, end-to-end, listen attend and spell architecture, OOV.**

I. Introduction

END-TO-END (e2e) techniques [1]–[4] have taken over hybrid approaches [5] to automatic speech recognition (ASR). While it is possible to predict characters or phonemes as their output [1], longer units such as words [6] and byte pair encoding units (BPEs) [7] are also widely used to preserve the wider context information. One issue that a hybrid approach struggles with is dealing with out-of-vocabulary words (OOVs): if a word was not seen during the language model (LM) training or has an unknown pronunciation, it is not possible to recognize it. An e2e system can avoid the OOV issue by choosing characters or BPEs as targets, but if longer units like words [6] are predicted, the problem is back.

The most common way of approaching OOV recovery is by using two levels of granularities (words and subwords) [8]. In the classic GMM/HMM-based ASR system, the system backs-off to the subword level and produces the corresponding letter-based output whenever the word-based ASR emits the OOV token (e.g. [9]). This backing-off has also been successfully applied in an e2e setup, for example in the context of connectionist temporal classification (CTC) [10]. Sometimes, instead of two granularities, a mix of word and sub-word units is used at the same level, for example, as LM units [11] or the neural network (NN) output for CTC [12]. In the field of natural language processing (NLP), a two-level word plus speller generative LM [13] has proven to work well for tasks involving OOVs. In [13], the open-vocabulary LM consists of two recurrent neural networks (RNNs): the first one captures the sentence structure, and the second one, called the speller, captures the word structure. The speller can generate new word types following the spelling style of in-vocabulary words (IVs). The novel words generated by this model fit the grammatical sentence structure well, but otherwise the model can produce a range of possible spellings that fit the language in question.

We propose to extend this approach to ASR and implement word-predicting e2e ASR training with a speller-like network. While the speller trained for LM tasks in [13] had only a text input to train on, ASR has the benefit of providing the speller with acoustic information too. This approach can potentially recover OOVs that are not only plausible from the LM point of view, but also acoustically correct. This training should also benefit IV representations by forcing the word embeddings within the ASR system to learn character representations as the second objective.

Our motivation for devising a more complex joint word-subword training architecture instead of continuing within the prevailing trend of using BPEs is the following:

1) Words and characters are linguistically motivated units of speech unlike rather ad-hoc BPEs, and a system predicting words and characters has the benefit of explainability.
2) A word-based system allows for direct application of external word-based LMs.
3) Pronunciation-aware representations (e.g. embeddings) of both IVs and OOVs obtained during the training can be used in other downstream (e.g. NLP) tasks that work with word-specific representations. One of such tasks is recovery of OOVs.

The novelty of the paper lies in jointly training the word predicting network (WPN) and the speller instead of working with two separate ASRs with outputs of different granularity. The benefit of this approach is training spelling-aware word embeddings that are better both for the ASR task and for OOV recovery.

II. Data

The experiments have been conducted on the well-know LibriSpeech dataset [14] of read speech. The training data contains two sets of clean speech, 100 hours and 360 hours, and a set of "other" of 500 hours. There are four separate sets for evaluation: "clean" and "other" development and test sets. Each of the four evaluation sets is about 5 hours long.

Manuscript received 25 March 2022; revised 17 June 2022; accepted 4 July 2022. Date of publication 19 July 2022; date of current version 11 August 2022. This work was supported in part by Horizon 2020 under Grant 870930 – WELCOME, in part by GAČR Project "NEUREM3" under Grant 19-26934X, and in part by Multi-Linguality in Speech Technologies under Grant LTAIN19087. Computing on IT4I supercomputer was supported by "e-Infrastructure CZ under Grant LM2018140. The associate editor coordinating the review of this manuscript and approving it for publication was Mr. Ville M. Hautamäki. *(Corresponding author: Ekaterina Egorova.)*

The authors are with the Faculty of Information Technology, Speech@FIT, Brno University of Technology, 601 90 Brno-střed, Czech Republic (e-mail: iegorova@fit.vutbr.cz; harivydana@gmail.com; burget@fit.vutbr.cz; cernocky@fit.vutbr.cz).

Digital Object Identifier 10.1109/LSP.2022.3192199

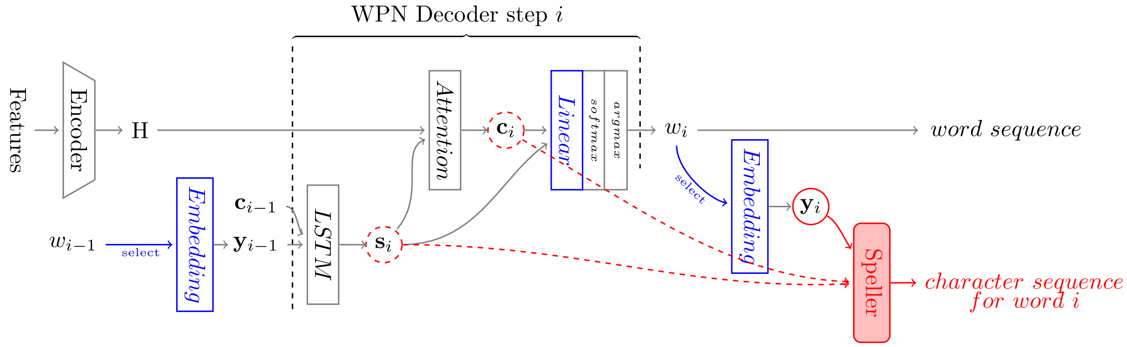

Fig. 1. Baseline LAS-like word-predicting network (WPN) (in black) and speller network trained on different inputs (in red). Weights of layers in blue are tied.

TABLE I
BASELINES OF THE 5000 BPE SYSTEM AND WPN SYSTEMS WITH DIFFERENT NUMBERS OF WORD TARGETS: 5000 AND 10000

system	d_clean	d_other	t_clean	t_other
5000BPE [18] no LM	4.87	14.37	4.87	15.39
5000BPE WER1	4.99	15.18	5.02	15.65
5000BPE rOOVs	63.8%	36.5%	62.0%	33.2%
5000BPE rIVs	95.7%	85.0%	95.6%	84.7%
5000w WER1	15.38	26.75	16.05	27.24
5000w WER2	6.47	19.12	6.92	19.43
10000w WER1	14.21	26.61	14.58	27.19
10000w WER2	8.66	21.78	8.79	22.36

LibriSpeech data is widely used for ASR experiments and numerous improvements have been reported on it recently. They often come from using external data (such as Libri-Light dataset, external LMs, etc.), more complex architectures (Conformers [15] etc.), significantly bigger models (1B parameters for w2v-BERT XXL [16] and SpeechStew [17]) or a combination of the above. To keep our model manageable, we have chosen to compare our baselines to the results in [18], as it uses a similar architecture (LAS) and model size (90 M parameters) and does not use external data.

III. E2E Architecture and Baselines

The baseline e2e ASR system used in this work is a LAS encoder-decoder model [4]. LAS provides complex and well-interpretable inner representations useful for joint training of tasks that benefit from sharing information. The input is log Mel-filterbank features of size 83. As for the outputs, we trained a system to predict 5000 BPE targets just to compare our architecture performance with that in [18]. In all the speller experiments, we work with word-predicting networks (WPN). The target vocabulary sizes that we experiment with are 5000 words (11.3% OOV rate) or 10000 words (6.6% OOV rate). We use SentencePiece toolkit [19] to obtain different tokenizations. In the word systems, OOVs have a special label and thus are represented by a dedicated embedding.

The input feature sequence $\mathbf{X}$ is transformed into the encoded hidden representation $\mathbf{H}$ by the **encoder**. The encoder consists of six layers, each containing a bi-directional long short-term memory (biLSTM) layer [1st 83 $\times$ 800, others 800 $\times (800+800)$] followed by a linear projection layer [1600 $\times$ 800]. The input sequence is sub-sampled in the time dimension by a factor of 2 in the first two encoder layers; the sub-sampling is done by maxpooling with the kernel size 3 and stride 2. Encoder layers have residual connections, and dropout is applied to the outputs of the biLSTM networks. Dropout is 0.1 in the subsampling layers and 0.3 in other layers.

The task of the **decoder** is to predict word labels from the hidden representation $\mathbf{H}$ [800 $\times$ (frames/4)]. Let w_i and w_{i-1} be the present and the past predicted word labels, while $\mathbf{s}_i$ [vector of size 800] is the state/output of the decoder LSTM:

$$\mathbf{s}_i = LSTM(\mathbf{s}_{i-1}, \mathbf{c}_{i-1}, \mathbf{y}_{i-1}), \tag{1}$$

and $\mathbf{c}_i$ [vector of size 800] is the context vector from the attention module:

$$\mathbf{c}_i = Attention(\mathbf{s}_i, \mathbf{H}). \tag{2}$$

Then, the current output label w_i is predicted by the LAS decoder as follows:

$$P(w_i|\mathbf{X}; w_1, .., w_{i-1}) = \text{argmax}(softmax(Linear([\mathbf{s}_i, \mathbf{c}_i]))). \tag{3}$$

One of the decoder LSTM inputs $\mathbf{y}_{i-1}$ [vector of size 1600] is the embedding vector of word label w_{i-1}. *Embedding* box in Fig. 1 denotes a linear layer that projects word label into its embedding of size 1600. Note that the weights of this layer are tied [20] to the weights of the linear layer from (3): $Embedding = Linear^T$. Thus, for each word label w_i, the embedding $\mathbf{y}_i$ is the corresponding row from the linear predicting layer weight matrix. OOV label is also represented by an embedding.

The model is trained using the ADAM [21] optimizer with the initial learning rate of 0.001, which is halved upon encountering an increase in the validation error rate. We do early stopping when observing an increase in the validation accuracy for three epochs. In the beginning of the training, 30000 steps of gradual warmup [22] are used. We perform scheduled sampling [23] where the correct label (teacher forcing) is selected with the probability 0.6 and the rest of the time the predicted (most likely) label is selected. This selected label w_i is then used for retrieving its embedding $\mathbf{y}_i$ to serve as the decoder LSTM input for the next time step.

The performances of our systems with different target vocabularies are presented in Table I. Our BPE-generating system WER

(5000BPE WER1) comes close to that of our chosen reference system (5000BPE [18] no LM), which is a reasonable baseline for our conditions. It is impossible for a WPN to generate words that were not in the training vocabulary. Thus, predicting the OOV label always causes an error in the classical word error rate (WER) calculation that we call WER1. We want to isolate the errors that happen due to the appearance of OOVs in a WPN and therefore we introduce WER2 scoring that treats the OOV label as a word. In terms of WER2, when a system predicts the OOV label for a reference word that is an OOV with the current vocabulary, there is no error. The differences between WER1 and WER2 for different vocabularies in Table I show that many of WER1 errors are due to vocabulary limitation. With bigger vocabularies, this difference is smaller, as the OOV rate is smaller as well. As we will be comparing our OOV recovery with the BPE-predicting system, we provide an analysis of BPE recovery potential. Lines named "BPE rOOVs" and "BPE rIVs" show percentages of correctly recovered rare (OOVs) and common (IVs) words. There are no OOVs in a BPE system, so we count the most frequent 10000 words (same as in the 10000 w system) as common words, and report their recovery rate in line "BPE rIVs". Rare words are words which are OOVs in the 10000 w system and their recovery rate is reported in line "BPE rOOVs". The analysis shows that there is potential for improvement in the task of recovering rare words in a BPE system.

IV. Speller Architectures

The speller consists of a single layer LSTM and a linear output layer, and it is optimized for predicting a string of characters that constitute a word. It solves a simpler problem, and so is much smaller than the WPN model: our baseline WPN model with 5000 outputs contains 90 M trainable parameters; the same model with the addition of the speller has 98 M trainable parameters.

A. Embedding Speller

Inspired by the LM-speller in [13], the first speller architecture we trained takes word embeddings $\mathbf{y}_i$ as an input and generates letters as an output. The limitation of this approach is that a number of diverse OOVs are assigned a single OOV label and thus a single embedding. This single embedding cannot learn all the spellings of OOVs, and therefore the OOV embedding is not updated through the speller. To spell an IV word, its embedding is repeated as a constant input for every output letter (one-to-many RNN). There is interaction between the WPN and the speller as the word embeddings are shared by both networks and are updated by backpropagating from both objectives during the training.

To accommodate updates from both networks, the training iterates between updating the WPN and the speller network. We have found it sufficient to update the speller once for each word in the vocabulary after every 500 mini-batches of 20 utterances of WPN training. Thus, each word contributes with an equal weight to updating the speller weights and also its embedding (WPN weights are not affected by the updates from the speller training step). There is a separate learning rate scheduling for speller updates, also with halving and warmup.

This training ensures that the word embeddings are trained to represent not only the information useful for word prediction but also the information useful for the speller task, i.e. it makes word embeddings spelling-aware. Such embeddings are beneficial for WPN training, but this approach does not allow for OOV recovery.

B. Context- and Acoustics-Aware Speller

To give the speller more information about the OOVs that we want it to spell, several other speller inputs have been tested (see dashed red lines in Fig. 1):

1) concatenation of word embedding $\mathbf{y}_i$ and context vector $\mathbf{s}_i$ from decoder LSTM,
2) concatenation of word embedding $\mathbf{y}_i$ and attention output $\mathbf{c}_i$,
3) concatenation of $\mathbf{y}_i$, $\mathbf{c}_i$, and $\mathbf{s}_i$.

Concatenating the word embedding with $\mathbf{s}_i$ gives the speller context information, as decoder LSTM preserves label history, while concatenating the word embedding with $\mathbf{c}_i$ gives the speller knowledge about acoustic information relevant for the currently decoded word.

As the speller needs current inner representations from the WPN, the training cannot proceed iteratively as for the embedding-only architecture. This is why the speller is updated simultaneously with the WPN during the training. After every word hypothesis is generated by the WPN, the speller is given the embedding $\mathbf{y}_i$ of the current label together with the current $\mathbf{c}_i$ and/or $\mathbf{s}_i$. Unlike during the embedding speller training described in subsection IV-A, here each embedding is not updated equal number of times during training, but the amount of times it appears in the training data, so the speller is better attuned to more frequent words. The speller is trained using cross-entropy loss to predict the correct sequence of characters (spelling) given the speller input. The reference character sequence is obtained by tokenizing the correct word label into characters. The overall loss for joint training of the WPN and the speller is a weighted combination of the two cross entropy losses (one for predicting words and one for spellings). For the experiments presented in Table II, equal cost weights are given to the speller and the WPN updates. In the test time, the WPN predicts a word label first, and if the predicted word label happens to be the OOV label, the speller network predicts the spelling of the current OOV from the current input. This cascade is thus able to both recognize IVs as well as recover OOVs. The output is a string of words: part of them are IVs and part are speller-generated.

V. Results and Analysis

Table II summarizes the results of the experiments with different speller architectures for a system with the vocabulary size 10000. The first two lines repeat baseline results in terms of WER1 and WER2 for the convenience of comparison (see section III for the explanation of WER1 and WER2). The rest of the systems are trained with a speller. The speller inputs of these systems are specified in the column "system".

TABLE II

Experiments With Different Speller Architectures on the 10000 Word System. WER1 and WER2 Show the Performance of the WPN Without Speller Participation. WERr Shows Results With OOVs Recovered Through the Speller, and rOOVs Shows the Percentage of OOVs That Were Recovered

system	metric	d_clean	d_other	t_clean	t_other
baseline	WER1	14.21	26.61	14.58	27.19
	WER2	8.66	21.78	8.79	22.36
$\mathbf{y}_i$	WER1	11.56	23.53	11.80	24.05
	WER2	8.05	20.82	8.61	21.10
	spell IV	12.84	24.75	13.33	25.13
[$\mathbf{y}_i$,$\mathbf{s}_i$]	WER1	10.79	21.56	10.98	22.07
	WER2	5.69	17.50	5.98	17.74
	WERr	11.39	22.39	11.36	22.76
	rOOVs	12%	5.2%	12.2%	5.1%
[$\mathbf{y}_i$,$\mathbf{c}_i$]	WER1	10.94	21.53	10.85	22.40
	WER2	5.82	17.39	5.80	18.07
	WERr	8.84	20.59	8.79	21.42
	rOOVs	32%	14.8%	32.3%	13.3%
[$\mathbf{y}_i$,$\mathbf{s}_i$,$\mathbf{c}_i$]	WER1	10.65	21.23	10.95	21.89
	WER2	5.49	17.15	5.95	17.54
	WERr	8.77	20.90	8.75	20.93
	rOOVs	32.4%	15.4%	34.6%	13.5%

A. Embedding Speller Results

The second section of Table II shows the results for the architecture and the training described in subsection IV-A, in which the speller gets only the word embedding $\mathbf{y}_i$ as an input. Note that this system uses only a single embedding representing any OOV and therefore it cannot recover spelling for OOVs. It can be seen that spelling-aware embeddings improve both WER1 and WER2. As the speller is not used in the decoding, the improvement in WER1 and WER2 does not come from the slight increase of the number of parameters, but solely from the fact that word embeddings are forced to be aware of the spelling.

The third metric ("spell IV") shows the performance of the speller. First, the word label is predicted, and if it is an IV label, it is passed through the speller to obtain character representation. Only if the spelling is correct, the word is not considered an error. The "spell IV" WER increases only slightly in comparison to WER1, which shows that the speller part learns to spell in-vocabulary embeddings almost perfectly.

B. Context- and Acoustics-Aware Speller Results

The last three sections of Table II show performances of the speller architectures with different inputs introduced in subsection IV-B. As before, WER1 and WER2 are scored on the output of the WPN and do not use the speller network during decoding. Any improvements happening with WER1 and WER2 in comparison with the previous system come due to the regularizing effect that the speller has on the word embeddings. The best improvement is reached with the speller input being the concatenation of $\mathbf{y}_i$, $\mathbf{c}_i$, and $\mathbf{s}_i$.

The two new metrics introduced for the speller systems show the capacity of these systems to recover OOVs through spelling. First, the word label is predicted, and if it is the OOV label, the inputs from the current decoding step are passed through the speller to obtain the character representation. The resulting output is then a sequence of words, some of them predicted IV words, and some recovered OOVs. This output is then scored against the reference transcription to obtain recovery WER – "WERr" metric. Meanwhile, "rOOVs" shows the percentage of OOVs that were ideally recovered through this process.

Concatenating word embedding $\mathbf{y}_i$ with $\mathbf{s}_i$ as the speller input makes the speller context-aware. This architecture has the drawback of not being able to spell an OOV if it happens to be the first word in a sentence: the input to the decoder LSTM is a vector of zeros. Table II clearly shows that the addition of the context information is not as helpful for WERr as the other architectures. However, this training still improves WER1 and WER2, and it is sometimes better at improving WER1 and WER2 than the $[\mathbf{y}_i, \mathbf{c}_i]$ speller architecture. As $\mathbf{c}_i$ contains representation of the acoustics relevant to the current word, concatenating the word embedding $\mathbf{y}_i$ with $\mathbf{c}_i$ makes the speller acoustics-aware. This information proves to be vital for OOV recovery, as is illustrated by the dramatically improved WERr and rOOVs scores. The speller system that takes the concatenation of $\mathbf{y}_i$, $\mathbf{s}_i$ and $\mathbf{c}_i$ as an input seems to take the best from both worlds and shows improvements across all metrics.

C. OOV Recovery Results

For every speller system, rOOVs shows the percentage of reference OOVs that were ideally recovered after speller decoding. We are able to reach 32-34% rOOVs for clean data and 13-15% on other. This can be directly compared to the system with BPE targets. Although the BPE system does not have OOVs per se, we score rOOVs on the words that are OOVs in the 10000 word vocabulary system. Table I shows 62-64% rOOVs in a BPE system for clean data and 33-36% on other.

While our numbers do not reach the recovery rates of the BPE system, our double-granularity system provides additional information about the word being an OOV and also its useful internal representation in the form of the speller input. Speller output does also better than BPEs for words that are not easily reproducible from common morphemes. For example, the speller correctly recovered the words "amputation" and "adventuring" whereas the BPE suggested "amutaion" and "adventureing". However, in general, BPE recovery performance is still better if one only cares about improving WER.

VI. Conclusion

In our work, we have proposed a new neural architecture for the ASR task that jointly trains two networks predicting both words and characters using shared inner representations. We have shown that forcing embeddings of a word-predicting system to also be spelling-aware improves WER of the word predicting task. Different inputs to the speller network have been tested, and we have shown their capability of recovering OOVs through spelling.

In the future, we would like to experiment with different training schedules and optimization metrics, to explore the benefits of having multiple embeddings to represent OOVs, and ultimately to reach and surpass the performance of BPE systems.

REFERENCES

[1] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 577–585.

[2] K. Audhkhasi, B. Ramabhadran, G. Saon, M. Picheny, and D. Nahamoo, "Direct acoustics-to-word models for English conversational speech recognition," in *Proc. Interspeech*, 2017, pp. 959–963.

[3] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid CTC/Attention architecture for end-to-end speech recognition," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 8, pp. 1240–1253, Dec. 2017.

[4] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, Attend and spell: A. neural network for large vocabulary conversational speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2016, pp. 4960–4964.

[5] M. Mohri, F. Pereira, and M. Riley, *Speech Recognition With Weighted Finite-State Transducers*. Berlin, Germany: Springer, 2008, pp. 559–584.

[6] H. Soltau, H. Liao, and H. Sak, "Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition," in *Proc. Interspeech*, 2017, pp. 3707–3711.

[7] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 1715–1725.

[8] A. Yazgan and M. Saraclar, "Hybrid language models for out of vocabulary word detection in large vocabulary conversational speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2004, pp. I–745.

[9] S. Kombrink, M. Hannemann, and L. Burget, "Out-of-vocabulary word detection and beyond," in *Proc. Eur. Conf. Mach. Learn. Princ. Pract. Knowl. Discov. Databases*, 2010, pp. 1–8.

[10] J. Li, G. Ye, R. Zhao, J. Droppo, and Y. Gong, "Acoustic-to-word model without OOV," in *Proc. IEEE Autom. Speech Recognit. Understanding Workshop*, 2017, pp. 111–117.

[11] M. A. B. Shaik, A. El-Desoky Mousa, R. Schlüter, and H. Ney, "Hybrid language models using mixed types of sub-lexical units for open vocabulary german LVCSR," in *Proc. Interspeech*, 2011, pp. 1441–1444.

[12] J. Li, G. Ye, A. Das, R. Zhao, and Y. Gong, "Advancing acoustic-to-word CTC model," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 5794–5798.

[13] S. Mielke and J. Eisner, "Spell once, summon anywhere: A. two-level open-vocabulary language model," *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 33, pp. 6843–6850.

[14] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2015, pp. 5206–5210.

[15] A. Gulati et al., "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, 2020, pp. 5036–5040.

[16] Y.-A. Chung et al., "w2v-BERT: Combining contrastive learning and masked language modeling for self-supervised speech pre-training," *Proc. IEEE Autom. Speech Recognit. Understanding Workshop*, 2021, pp. 244–250.

[17] W. Chan, D. S. Park, C. Lee, Y. Zhang, Q. V. Le, and M. Norouzi, "SpeechStew: Simply mix all available speech recognition data to train one large neural network," *Proc. MLSLP*, 2021.

[18] A. Zeyer, K. Irie, R. Schlüter, and H. Ney, "Improved training of end-to-end attention models for speech recognition," in *Proc. Interspeech*, vol. abs/1805.03294, 2018, pp. 7–11.

[19] T. Kudo and J. Richardson, "SentencePiece: A. simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proc. Conf. Empirical Methods Natural Lang. Process.: Syst. Demonstrations*, 2018, pp. 66–71.

[20] H. Inan, K. Khosravi, and R. Socher, "Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling," in *Proc. Int. Conf. Learn. Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=r1aPbsFle

[21] P. Diederik Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.

[22] P. Goyal et al., "Accurate, large minibatch SGD: Training imagenet in 1 hour," vol. abs/1706.02677, 2017, *arXiv:1706.02677*.

[23] S. Bengio, O. Vinyals, N. Jaitly, and N. M. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1171–1179.