

HIERARCHICAL SOFTMAX FOR END-TO-END LOW-RESOURCE MULTILINGUAL SPEECH RECOGNITION

Qianying Liu^{1*}, Zhuo Gong^{3*}, Zhengdong Yang^{1*}, Yuhang Yang², Sheng Li⁴, Chenchen Ding⁴,
Nobuaki Minematsu³, Hao Huang², Fei Cheng¹, Chenhui Chu¹, Sadao Kurohashi¹

¹Graduate School of Informatics, Kyoto University, Sakyo-ku, Kyoto, Japan

²School of Information Science and Engineering, Xinjiang University, Urumqi, China

³Department of EEIS, Graduate School of Engineering, The University of Tokyo, Tokyo, Japan

⁴National Institute of Information and Communications Technology (NICT), Kyoto, Japan

ABSTRACT

Low-resource speech recognition has been long-suffering from insufficient training data. In this paper, we propose an approach that leverages neighboring languages to improve low-resource scenario performance, founded on the hypothesis that similar linguistic units in neighboring languages exhibit comparable term frequency distributions, which enables us to construct a Huffman tree for performing multilingual hierarchical Softmax decoding. This hierarchical structure enables cross-lingual knowledge sharing among similar tokens, thereby enhancing low-resource training outcomes. Empirical analyses demonstrate that our method is effective in improving the accuracy and efficiency of low-resource speech recognition.

Index Terms: Speech recognition, Acoustic model, End-to-End Multilingual Model

1. INTRODUCTION

Automatic speech recognition (ASR) systems have gained remarkable progress in the past few years. Nevertheless, the present ASR systems cater to only approximately 100 out of the 7000 spoken languages worldwide. To address this limitation, multilingual models have garnered much attention. These models can learn universal features, transferable from resource-rich to resource-limited languages, and support multiple languages with a single ASR model. Early studies utilized context-dependent deep neural network hidden Markov models [1], which relied on hand-crafted pronunciation lexicons. However, when adapted to low-resource languages, such systems exhibit limitations due to the absence of sufficient modeling techniques. Attention-based end-to-end (E2E) models simplify training and eliminate the dependence on pronunciation lexicons [2, 3, 4]. Recent studies employing E2E models have focused on learning universal representations at the encoding stage, using transfer

learning techniques [5, 6, 7, 8, 9, 10], as well as on hierarchical embedding of phonemes, phones, and phonological articulatory attributes [11]. Meanwhile, large pretrained models [12, 13, 14, 15] and multilingual speech corpora [16, 17, 18, 19, 20] have been investigated for learning pre-trained representations.

In this paper, we aim to investigate the explicit transfer of cross-language knowledge at the decoding stage, which has been largely unexplored in prior studies that focus on encoder representations. Based on linguistic studies on the presence of similar modeling unit distributions in neighboring languages [21], we propose an efficient method to capture the similarity among these units, such as characters and globalphones, at the decoding stage. Our approach utilizes Huffman coding to automatically capture the similarity among modeling units, relying only on monolingual data. We introduce hierarchical Softmax (H-Softmax) [22], an approximation softmax inspired by binary trees, to model the similarity during decoding. This structure enables similar units to share decoder representations, thus improving model performance, and also breaks down the expensive softmax step into several binary classifications, thereby enhancing model efficiency [23]. We design a vectorization algorithm that can expedite the training and inference procedures, enabling our method to outperform the vanilla softmax on GPUs in terms of efficiency.

With the combination of the two components, our method:

- Automatically captures cross-lingual modeling unit similarity for multilingual ASR.
- Leverages H-Softmax to achieve higher efficiency and reduce computational complexity.

While previous studies have utilized H-Softmax in the neural language model of ASR [24, 25]. However, to the best of our knowledge, no existing work has investigated the direct application of H-Softmax in the acoustic model. Furthermore, our study is the first to explore the potential of H-Softmax for low-resource multilingual ASR.

* denotes equal contribution.

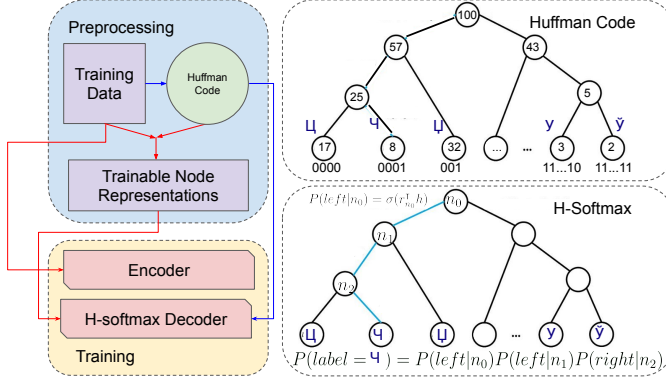


Fig. 1. The flowchart of the proposed method. The blue lines stands for determination relation, and the red line stands for how the data goes through the model.

2. PROPOSED METHOD

In this section, we introduce the proposed method of this paper, as shown in Fig 1. We first use the training text data to determine the Huffman code for each token as described in subsection 2.1. Then we build the ASR model, where an encoder takes in the source speech signals, and the decoder predicts the Huffman code of target text with H-Softmax. At the inference stage, the model predicts a sequence of post-processed Huffman codes to text.

2.1. Huffman Code

Based on the assumption that neighbour languages share similar token distribution, the concept of our proposed method is to generate a representation code for each token via frequency clustering. We first generate the Huffman code of each token. Formally, given the multilingual token vocabulary $V = \{t_1, t_2, \dots, t_N\}$, where N is the vocabulary size, we maintain the term frequency set $S_p = \{p_{t_i}\}_{i=1}^N$, where the same token in different languages are considered as one token. With S_p , we generate a Huffman tree of $V = \{t_i\}$ via frequency clustering and further recursively generate the Huffman code by assigning 0 to the left subtree and 1 to the right subtree.

2.2. Model Architecture

For the ASR model, we use conformer [26] as the encoder and transformer [27] as the decoder. For the decoder, we replace the vanilla softmax with H-Softmax.

H-Softmax organizes the output vocabulary into a tree where the leaves are the vocabulary tokens, and the intermediate nodes are latent variables [22]. We use the Huffman tree generated in subsection 2.1 as the tree for H-Softmax that there is a unique path from the root to each token, forming a complete binary tree. We follow [22] and apply the binary

tree H-Softmax. The decoding procedure is transformed into predicting one leaf node of the binary tree at each timestep. Each leaf node, which represents a token, could be reached by a path from the root through the inner nodes. Given the transformer output hidden state h and trainable node representation vectors $\{r_i\}$, the final possibility of a leaf node w_i could be represented as:

$$P(\text{label} = w_i) = \prod_{\text{path}} P(\text{path}_k | n_k) \quad (1)$$

$$= \prod_{\text{path}} \begin{cases} \sigma(r_k^T h) & \text{if } \text{path}_k = \text{left}, \\ 1 - \sigma(r_k^T h) & \text{if } \text{path}_k = \text{right}. \end{cases}$$

where n_k stands for the k th node on the path from the root to w_i such as n_0, n_1 and n_2 in Fig. 1; path_k stands for the branch leading towards w_i which is *left* or *right*, and $\sigma(\cdot)$ stands for sigmoid function.

2.3. Efficient Implementation of H-Softmax

While decomposing the Softmax to a binary tree H-Softmax reduces the decoding time complexity from $O(V)$ to $O(\log(V))$, and the train time complexity remains $O(V \log(V))$. Since previous H-Softmax implementation[23] is on CPU, considering the order of magnitude difference between CPU's and GPU's FLOPs, the challenge of improving the efficiency of model training lies in designing an implementation of H-Softmax for GPU training. We propose a vectorization algorithm to accelerate training and decoding procedures on GPU.

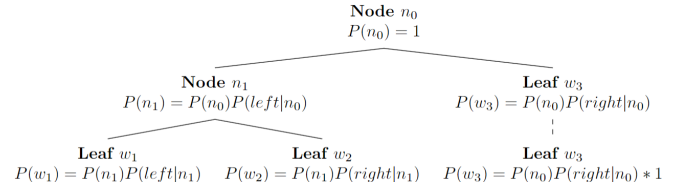


Fig. 2. A typical H-Softmax tree structure. Leaf w_3 has a virtual child with the same probability of aligning each leaf node to the same depth, so it is conceptually possible for path vectorization.

To explain the core idea of our vectorization algorithm, we show a typical H-Softmax tree structure in Fig 2. Then, we can vectorize log-probability calculations from Eq 1 to the followings:

$$\begin{bmatrix} \log P(w_1) \\ \log P(w_2) \\ \log P(w_3) \end{bmatrix} = \begin{bmatrix} \log[\sigma(r_1^T h) \sigma(r_2^T h)] \\ \log[\sigma(r_1^T h) (1 - \sigma(r_2^T h))] \\ \log[1 - \sigma(r_1^T h)] \end{bmatrix}$$

$$= \sum_{\text{column}} \log \begin{bmatrix} 1 * \sigma(r_1^T h) + 0 & 1 * \sigma(r_2^T h) + 0 \\ 1 * \sigma(r_1^T h) + 0 & -1 * \sigma(r_2^T h) + 1 \\ -1 * \sigma(r_1^T h) + 1 & 0 * \sigma(r_2^T h) + 1 \end{bmatrix}$$

$$= \sum_{\text{column}} \log(\text{Sign} \circ \sigma(p) + \text{Bias})$$

where Sign is a 3 by 2 matrix of σ 's signs, Bias is a 3 by 2 matrix of σ 's biases and p is the result vector of the in-

ner product between node vectors and h . After building the Huffman tree, the *Sign* and *Bias* matrices are fixed. So in the training stage, leaf node log probabilities can be acquired only by vector operations.

For decoding, we only need leaves with the highest probabilities. To directly calculate this objective, different from training, we also develop a path-encoding-based multi-layer beam searching on GPU for H-softmax¹ to retain the time efficiency advantage of time-space complexity $O(\log(V))$ compared to vanilla softmax’s $O(V)$.

3. EXPERIMENT EVALUATIONS

In this section, we evaluate the proposed method in two low-resource settings. To examine the effectiveness of our method in more extensive settings, specifically in the same language group and cross-language groups, we first simulate a low resource setting by down-sampling from a speech-to-text multilingual large-scale dataset. To further verify the performance on natural low-resource languages and other token modeling units, we test our method on an extremely low-resourced zero-shot cross-lingual speech-to-phoneme setting. For the high-resource setting, where every token could be fully trained, modeling of neighbors could no longer be useful, and thus our experiments focus on the low-resource setting.

3.1. Data Description

We sample our speech-to-text datasets from Common Voice Corpus 11.0. We selected three different linguistic groups: Romance, Slavic, and Turkic. For each group, we selected five languages from the corpus and constructed training and testing sets with the validated data of each language. With the data size of many existing datasets being around 20~30 hours [28, 29], we downsampled the training data to the extent of 30 hours per language on average to simulate a low-resource setting. As a result, the total size of training data changed from 5492 hours to 450 hours, with the general downsampling ratio $\lambda = 0.082$. Different downsampling ratios are used for different languages to counter the imbalance of data size among languages. For a set of languages $\{L_1, \dots, L_m\}$ with their proportion $\{p_1, \dots, p_m\}$, the down-sampling ratio for language L_i is obtained by $\lambda_i = \frac{p_i^{\alpha-1}}{\sum_{j=1}^m p_j^{\alpha}} \lambda$, where α is a smooth coefficient that we set to 0.5. The size of testing sets is the lesser of 10% of the validated data and 10,000 utterances.

For speech-to-phoneme datasets, we use UCLA Phonetic Corpus [19]. The Edo-Ancient Tokyo (bin), Kele-Congo (sbc), Malayalam-Malay (mal), Creole-Cape Verde (kea), Klao-Liberia (klu), Arabic-Tunisian Spoken (aeb), Makasar-South Sulawesi (mak), Finnish-Finland (fin), Abkhaz-North Caucasian (abk) and Aceh-Sumatra (ace) are testing languages and other languages are used for training.

Table 1. Speech-to-text datasets statistics.

Group	Language	Training (Hours)	Testing (Hours)
Romance	Catalan (ca)	76.3	15.3
	Spanish (es)	37.6	14.4
	French (fr)	55.7	13.4
	Italian (it)	33.4	15.0
	Portugal (pt)	20.7	11.4
Slavic	Belarusian (be)	63.0	13.9
	Czech (cs)	42.5	5.8
	Polish (pl)	37.8	12.3
	Russian (ru)	27.3	14.5
	Ukrainian (uk)	23.4	7.2
Turkic	Bashkir (ba)	29.6	12.6
	Kyrgyz (ky)	11.3	3.8
	Turkish (tr)	16.9	8.4
	Tatar (tt)	10.0	3.0
	Uzbek (uz)	18.1	10.4

Table 2. Speech-to-phoneme datasets.

Language	Dataset	Hours
UCLA Phonetic Corpus (97 languages)	Training (87 lang.)	1.86
	Testing (10 lang.)	0.14

3.2. Model Training

For acoustic features, the 80-dimensional log-Mel filterbanks (FBANK) are computed with a 25ms window and a 10ms shift. Besides, SpecAugment [30] is applied to 2 frequency masks with maximum frequency mask ($F = 10$) and 2 time masks with maximum time mask ($T = 50$) to alleviate overfitting.

Both H-Softmax and Softmax models are trained using the same network structure. The networks are constructed using WeNet toolkit [31].¹ Two convolution sub-sampling layers with kernel size 3×3 and stride 2 are used in the front of the encoder. For model parameters, we use 12 conformer layers for the encoder and 6 transformer layers for the decoder.

Adam optimizer is used with a learning rate schedule with 25,000 warm-up steps. The initial learning rate is 0.00005. 100 max epochs for speech-to-text datasets and 80 max epochs for speech-to-phoneme datasets.

3.3. Speech-to-text Recognition Evaluation

We conducted two sets of experiments on speech-to-text datasets: training with languages within one language group and training with languages across language groups. We tokenized the transcriptions at the character level as we verified that it outperforms tokenizing in the sub-word level (such as BPE).² As shown in Table 4, when training with languages

¹Our implementation is <https://github.com/Derek-Gong/hsoftmax>

²Preliminary experiments on Slavic language group with traditional Softmax results in a CER of 8.5% for character-level tokenization, 8.9% for

Table 3. PER% on speech-to-phoneme datasets.

Model	bin	sbc	mal	kea	klu	aeb	mak	fin	abk	ace	Overall
Softmax	70.4	87.4	98.0	83.4	86.2	84.1	84.5	94.2	75.0	75.5	85.2
H-Softmax	38.0	68.9	80.8	59.6	62.9	60.9	73.7	75.2	70.9	48.6	64.7

Table 4. CER% on speech-to-text datasets. Models are trained with languages within the same language group.

Model	ca	es	fr	it	pt	Overall
Softmax	5.5	9.4	12.1	8.8	9.7	9.1
H-Softmax	5.3	8.8	11.3	8.4	10.0	8.8

Model	be	cs	pl	ru	uk	Overall
Softmax	5.0	5.4	9.2	11.6	11.5	8.5
H-Softmax	4.8	5.4	8.8	10.4	11.6	8.2

Model	ba	ky	tr	tt	uz	Overall
Softmax	16.9	18.3	14.0	10.2	20.6	16.0
H-Softmax	14.4	17.2	12.8	9.1	16.3	14.0

within the same language group, our Huffman code achieves better performance (character error rate, CER%) than traditional Softmax for all three groups, which demonstrates the effectiveness of our method. Results in Table 5 show that our method can also make improvements when training with the combined data in 15 languages across different language groups, showing that our method works in a more extensive range of scenarios than we expected. In addition, our method is more robust as there are no languages with a distinctively high error rate, unlike traditional Softmax.

3.4. Zero-shot Cross-lingual Speech-to-phoneme Recognition Evaluation

Our proposed model demonstrated superior performance compared to the conventional model (phone error rate, PER%) across all languages on the UCLA Phonetic corpus, as presented in Table 3. Overall, our approach outperformed the softmax baseline by a significant margin of 20.51% PER. On the ‘bin’ language, the performance gap between softmax and H-softmax was 32.33%. These results showcase the effectiveness of our method in automatically constructing universal representations for multilingual ASR and achieving zero-shot cross-lingual phoneme recognition.

3.5. Decoding Speed

We observed decoding acceleration with our proposed H-Softmax model in Table 6. Decomposing the Softmax output layer to a binary tree reduces the complexity of obtaining probability distribution from $O(V)$ to $O(\log(V))$, which leads to improvement in efficiency. The results also show that the

BPE (vocab size = 500) and 9.6% for BPE (vocab size = 5000).

Table 5. CER% on speech-to-text datasets. Models are trained with all the languages across language groups.

Model	ca	es	fr	it	pt	Overall
Softmax	7.3	9.4	15.1	8.4	21.8	12.4
H-Softmax	5.7	9.0	11.3	7.3	15.4	9.7

Model	be	cs	pl	ru	uk	Overall
Softmax	5.8	5.8	8.5	10.6	11.5	8.4
H-Softmax	6.1	5.9	8.2	8.5	9.8	7.7

Model	ba	ky	tr	tt	uz	Overall
Softmax	11.8	13.7	12.3	7.6	13.5	11.8
H-Softmax	11.0	13.6	10.3	7.1	13.7	11.1

Table 6. RTF (real-time factor) of the decoding process of Table 5. 3 languages are selected to show the decoding speed with different tokens per sentences (tok/sent).

Model	tr 32.6 tok/sent	pl 47.4 tok/sent	ru 63.1 tok/sent
Softmax	0.022	0.023	0.026
H-Softmax	0.018	0.018	0.020

sentence length (tokens per sentences) determines the decoding time difference between Softmax and H-softmax. Longer sentences take more time steps for inference, and the time difference between the two models exponentially increases.

4. CONCLUSION

This paper proposes an automatic method to generate cross-lingual representations, which facilitate the acquisition of cross-lingual universal features in neighboring languages. Our approach employs Huffman coding, which utilizes token frequencies (characters, globalphones, etc.) to bridge different languages. Furthermore, we introduce H-Softmax, which improves model performance by enabling similar units that share Huffman binary representations and accelerates decoding compared to traditional Softmax methods. As future work, we aim to design binary codes that incorporate not only shallow frequency terms but also more semantically meaningful features, such as token embeddings.

5. ACKNOWLEDGEMENTS

This study was supported by JST SPRING Grant No.JPMJSP2110 and Grant-in-Aid for Scientific Research (C) No. 23K11227.

6. REFERENCES

- [1] G. Dahl, D. Yu, L. Deng, and A. Acero, “Context dependent pre-trained deep neural networks for large vocabulary speech recognition,” *IEEE Trans. ASLP*, vol. 20, no. 1, pp. 30–42, 2012.
- [2] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and et al., “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.
- [3] A. Graves and N. Jaitly, “Towards end-to-end speech recognition with recurrent neural networks,” in *Proc. ICML*. PMLR, 2014, pp. 1764–1772.
- [4] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio, “End-to-end attention-based large vocabulary speech recognition,” in *Proc. IEEE-ICASSP*. IEEE, 2016, pp. 4945–4949.
- [5] X. Li and et al., “Towards zero-shot learning for automatic phonemic transcription,” in *Proc. AAAI*, 2020.
- [6] C. Zhu and et al., “Multilingual and crosslingual speech recognition using phonological-vector based phone embeddings,” in *Proc. ASRU*, 2021.
- [7] Sheng Li, Chenchen Ding, Xugang Lu, Peng Shen, Tatsuya Kawahara, and Hisashi Kawai, “End-to-end articulatory attribute modeling for low-resource multilingual speech recognition,” in *INTERSPEECH*, 2019, pp. 2145–2149.
- [8] W. Hou and et al., “Exploiting adapters for cross-lingual low-resource speech recognition,” 2021.
- [9] W. Hou and et al., “Large-scale end-to-end multilingual speech recognition and language identification with multi-task learning,” in *Proc. INTERSPEECH*, 2020.
- [10] D. Liu and et al., “Learning phone recognition from unpaired audio and phone sequences based on generative adversarial network,” 2021.
- [11] X. Li, J. Li, F. Metze, and A. Black, “Hierarchical phone recognition with compositional phonetics,” in *Proc. Interspeech*, 2021, pp. 2461–2465.
- [12] A. Baevski and et al., “wav2vec2.0: A framework for self-supervised learning of speech representations,” in *Proc. NeurIPS*, 2020.
- [13] A. Baevski and et al., “Unsupervised speech recognition,” in *Proc. NeurIPS*, 2021.
- [14] Y. Wang and et al., “A fine-tuned wav2vec 2.0/hubert benchmark for speech emotion recognition, speaker verification and spoken language understanding,” in *arXiv preprint arXiv:2111.02735*, 2021.
- [15] S. Chen and et al., “Wavlm: Large-scale self-supervised pre-training for full stack speech processing,” in *arXiv preprint arXiv:2110.13900*, 2021.
- [16] C. Wang and et al., “Covost: A diverse multilingual speech-to-text translation corpus,” in *Proc. LREC*, 2020.
- [17] C. Wang and et al., “Covost 2: A massively multilingual speech-to-text translation corpus,” in *arXiv preprint arXiv:2007.10310*, 2020.
- [18] R. Ardila and et al., “Common voice: A massively-multilingual speech corpus,” in *Proc. LREC*, 2020.
- [19] Xinjian Li, David R. Mortensen, Florian Metze, and Alan W Black, “Multilingual phonetic dataset for low resource speech recognition,” in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 6958–6962.
- [20] A. Black, “Cmu wilderness multilingual speech dataset,” in *Proc. ICASSP*, 2019.
- [21] Mikel Artetxe, Gorka Labaka, and Eneko Agirre, “A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings,” in *Proc. ACL*, 2018.
- [22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in neural information processing systems*, 2013, pp. 3111–3119.
- [23] Abdul Arfat Mohammed and Venkatesh Umaashankar, “Effectiveness of hierarchical softmax in large scale classification tasks,” in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2018, pp. 1090–1094.
- [24] Tuka AlHanai, Wei-Ning, and James Glass, “Development of the mit asr system for the 2016 arabic multi genre broadcast challenge,” in *IEEE-SLT (Spoken Language Technologies Workshop)*, 2016.
- [25] Seppo Enarvi, Peter Smit, Sami Virpioja, and Mikko Kurimo, “Automatic speech recognition with very large conversational finnish and estonian vocabularies,” 2017.
- [26] A. Gulati, J. Qin, C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and et al., “Conformer: Convolution-augmented transformer for speech recognition,” *arXiv preprint arXiv:2005.08100*, 2020.
- [27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, “Attention is all you need,” in *NIPS*, 2017, pp. 5998–6008.
- [28] D. Wang and X. Zhang, “THCHS-30 : A free chinese speech corpus,” *CoRR*, vol. abs/1512.01882, 2015.
- [29] R. Aisikaer, S. Yin, Z. Zhang, D. Wang, H. Askar, and F. Zheng, “THUYG-20: A free uyghur speech database,” *Journal of Tsinghua University(Science and Technology)*, vol. 57(2): 182-187, 2017.
- [30] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin Dogus Cubuk, and Quoc V. Le, “SpecAugment: A simple augmentation method for automatic speech recognition,” in *INTERSPEECH*, 2019.
- [31] Zhuoyuan Yao, Di Wu, Xiong Wang, Binbin Zhang, Fan Yu, Chao Yang, Zhendong Peng, Xiaoyu Chen, Lei Xie, and Xin Lei, “WeNet: Production Oriented Streaming and Non-Streaming End-to-End Speech Recognition Toolkit,” in *Proc. Interspeech 2021*, 2021, pp. 4054–4058.

A. HUFFMAN CODING

Formally, given a set of m languages $\{L_i\}_{i=1}^m$ and the corresponding character sets $S^i = \{c_1^i, c_2^i, \dots, c_{V^i}^i\}$, where V^i is the character vocabulary size. For each language L_i individually, we maintain the term frequency $S_p^i = \{p_{c_j^i}\}$ of each character j in monolingual data. Term frequency is defined as follow:

$$p_{c_j^i} = \frac{f_{c_j^i}}{\sum_{j=1}^{V^i} f_{c_j^i}} \quad (2)$$

Where $f_{c_j^i}$ is the raw count of a term c_j^i in the monolingual data.

Algorithm 1 Generating the Huffman Tree of Characters

Require: S_p
Assign leaf nodes $\{N_{c_j^i}\}$ for elements in S_p ;
sort $\{N_{c_j^i}\}$ based on $\{p_{c_j^i}\}$ order to form a priority queue Q ;
while $sizeof(Q) \geq 2$ **do**
 Remove 2 lowest probability nodes N_a and N_b from Q ;
 Create a new internal node N_c with N_a and N_b as children, probability $p_c \leftarrow$ sum of p_a and p_b ;
 Add N_c to Q ;
end while
 $N \leftarrow$ last node in Q
return N

Each element in S_p is assigned a leaf node $N_{c_j^i}$ and then pushed into a priority queue Q . The priority is determined by the probability $p_{c_j^i}$, that the lower probability $p_{c_j^i}$ has higher priority in the queue Q . Then the algorithm generates Huffman tree by removing the 2 highest priority nodes N_a and N_b from Q and then merge them into a new node N_c , which takes the higher priority node of the two removed nodes as the left children and the other as the right children. The priority p_c is determined by the sum of p_a and p_b . We repeat this process until there is only one node N in Q , then N is the root node of the Huffman tree.

Algorithm 2 function $code(N, c)$, Huffman Coding of Characters

Require: $N, c \leftarrow None$
 $N_l, N_r \leftarrow$ children of N
if N_l is not leaf node **then**
 $\{code_{c_j^i}\} \leftarrow code(N_l, c + 0)$
else
 $code_{N_l} = c + 0$
end if
if N_r is not leaf node **then**
 $\{code_{c_j^i}\} \leftarrow code(N_r, c + 1)$
else
 $code_{N_r} = c + 1$
end if
return $\{code_{c_j^i}\}$

Given the Huffman tree root node N , as shown in Algorithm 2, recursively we generate the Huffman code $code_{c_j^i}$ for each leaf node $N_{c_j^i}$. The root node N starts with an empty representation c , for each node the code of its left children N_l is $c + 0$, and the code of its right children N_r is $c + 1$. We can recursively traverse the tree and give every leaf node a corresponding code $code_{c_j^i}$.

Algorithm 3 Arbitrary Binary Tree Based H-Softmax

Require: $N, vocab_size, depth, inner_size, hidden_size$
/*Preprocessing before training*/
for i from 0 to $inner_size - 1$ **do**
 non-leaf $node_i.index \leftarrow i$
end for
 $Index, Sign, Bias \leftarrow$ zero matrices of $(vocab_size, depth)$
for each $leaf_i \in N$ **do**
 $path_i \leftarrow$ the path from root N to $leaf_i$
 for each $node_j \in path_i$ **do**
 if $node_{j+1}$ is $node_j$'s left child **then**
 $Index_{ij} \leftarrow node_j.index$
 $Sign_{ij} \leftarrow 1$
 else if $node_{j+1}$ is $node_j$'s right child **then**
 $Index_{ij} \leftarrow node_j.index$
 $Sign_{ij} \leftarrow -1$
 $Bias_{ij} \leftarrow 1$
 end if
 end for
end for
/*Forward pass*/
 $h \leftarrow sigmoid(hidden_vecs * embedding)$
 $H \leftarrow$ stack h vertically $vocab_size$ times
 $H \leftarrow H$ is index selected by $Index$ in the last dimension
 $log_probs \leftarrow \sum_{column} log(Sign * H + Bias)$
return log_probs

B. H-SOFTMAX

In order to vectorize calculations on a binary tree (Fig 2), we need to vectorize paths from root to leaves first. For a path, every time it turns left or right, we multiply the probability of this path by $1 - \sigma(h)$ or $\sigma(h)$ in which h is inner product of current node's hidden vector and embedding vector fed into H-Softmax. When the path goes to its leaf node, the accumulated probability of a token is obtained. Thus, a complete path can be composed of three kinds of path elements: $1 - \sigma(h)$, $\sigma(h)$, or 1 which are corresponding to left node, right node, or padding node (dot line in Fig 2). We need the padding node to pad each path to depth of the tree. Finally, the accumulated probability is a product of all of the path elements.

Now, let's focus on these path elements. Remember that $\sigma(h)$ is a variable for each different sample, while path elements' sign (-1, 1, 0) and bias (1, 0, 1) is fixed because the tree structure is fixed. So, we can extract signs and bias of a path into two row vectors $path_sign_encoding_i$ and $path_bias_encoding_i$, then stack them vertically to matrices of $Sign$ and $Bias$. Meanwhile, we can stack

h from all non-leaf node to one column vector. Because of log operation, padding nodes which is expressed by sign 0 and bias 1 is automatically eliminated.

We present above algorithms in Algorithm 3. And we need to emphasize that in practice the intermediate matrix H is index selected so that its column number is reduced from $inner_size$ to $depth$. Since $depth$ is roughly $\log(vocab_size)$, this step is crucial to maintain the algorithm in time complexity of $O(vocab_size * hidden_size + vocab_size * \log(vocab_size))$ instead of $O(vocab_size * hidden_size + vocab_size^2)$ while it's $O(vocab_size * hidden_size)$ for vanilla softmax.