LIBRIHEAVY: A 50,000 HOURS ASR CORPUS WITH PUNCTUATION CASING AND CONTEXT

Wei Kang, Xiaoyu Yang, Zengwei Yao, Fangjun Kuang, Yifan Yang, Liyong Guo, Long Lin, Daniel Povey

Xiaomi Corp., Beijing, China

{kangwei1, dpovey}@xiaomi.com

ABSTRACT

In this paper, we introduce Libriheavy, a large-scale ASR corpus consisting of 50,000 hours of read English speech derived from LibriVox. To the best of our knowledge, Libriheavy is the largest freely-available corpus of speech with supervisions. Different from other open-sourced datasets that only provide normalized transcriptions, Libriheavy contains richer information such as punctuation, casing and text context, which brings more flexibility for system building. Specifically, we propose a general and efficient pipeline to locate, align and segment the audios in previously published Librilight to its corresponding texts. The same as Librilight, Libriheavy also has three training subsets small, medium, large of the sizes 500h, 5000h, 50000h respectively. We also extract the dev and test evaluation sets from the aligned audios and guarantee there is no overlapping speakers and books in training sets. Baseline systems are built on the popular CTC-Attention and transducer models. Additionally, we open-source our dataset creatation pipeline which can also be used to other audio alignment tasks.

Index Terms— Speech recognition, Corpus, Audio alignment, Librivox

1. INTRODUCTION

In the past decade, various system architectures, like Connectionist Temporal Classification (CTC) [1], RNN-T [2] and encoder-decoder based model [3], have been proposed, pushing the dominant framework from the hybrid Hidden Markov Models (HMM) [4] to end-to-end models. In general, the neural network models are supposed to be more data hungry than traditional systems.

A lot of work has been done on publishing open source datasets, for example, the Wall Street Journal corpus [5], SwitchBoard [6], Fisher [7] and the famous LibriSpeech corpus [8]. While these are all small or medium size datasets with less than 2,000 hours, which is too small to train a good enough end-to-end model. In recent years, there are also large-scale corpora like GigaSpeech [9], People's Speech [10] and MLS [11]. One drawback of these datasets is that they only provide normalized transcriptions, making it impossible to train a model that needs full-format texts, such as punctuation prediction.

Typical ASR corpora aim at training ASR systems to recognize independent utterances. However, the preceding context of the current utterance may convey useful information. Contextualized speech recognition utilizes the cross-utterance context to improve the accuracy of ASR systems and yields promising results [12, 13]. However, training such systems usually requires utterance-level context for each training utterance, which is not available in most existing ASR corpora. Therefore, such a dataset with textual context information is highly desirable.

Motivated by the aforementioned points, we introduce Libriheavy, a large-scale (50,000 hours) corpus containing not only fully formatted transcripts but also textual context, which is suitable for various speech recognition related tasks. In addition, unlike other open-source datasets that have their own creating pipelines, we propose a general audio alignment method and release it as a standard package. Our contributions are as follows:

- We release a 50,000-hour of labeled audio containing punctuation casing and preceding text;
- We propose and open-source a general audio alignment pipeline, which makes it easier to construct ASR corpora;
- We provide solid evaluation results on Libriheavy, which demonstrate the high quality of the corpus and the robustness of our pipeline.

2. LIBRIHEAVY CORPUS

In this section, we provide a detailed description of the Libriheavy corpus, including audio files, metadata, data partitions, text styles, and other aspects. Instructions and scripts are available in the Libriheavy GitHub repository[1].

2.1. Librilight

Librilight [14] is a collection of unlabeled spoken English audio derived from open-source audio books from the LibriVox project [2]. It contains over 60,000 hours of audio and aims for training speech recognition systems under limited or no supervision. The corpus is free and publicly available [3].

2.2. Libriheavy

Libriheavy is a labeled version of Librilight. We align the audio files in Librilight to their corresponding text in the orig-

---

[1] https://github.com/k2-fsa/libriheavy
[2] https://librivox.org
[3] https://github.com/facebookresearch/libri-light

Table 1: The dataset statistics of Libriheavy.

subset	hours	books	per-spk hrs	total spks
small	509	173	1.22	417
medium	5042	960	3.29	1531
large	50794	8592	7.54	6736
dev	22.3	180	0.16	141
test-clean	10.5	87	0.15	70
test-other	11.5	112	0.16	72

inal book and segment them into smaller pieces with durations ranging from 2 to 30 seconds. We maintain the original dataset splits of Librilight and have three training subsets (small, medium, large). In addition, we further extract evaluation subsets (dev, test-clean, test-other) for validation and testing. Table 1 shows the statitics of these subsets.

2.2.1. Metadata

We save the metadata of the dataset as Lhotse [15] cuts in JSON lines. Each line is a self-contained segment, including the transcript and its audio source. Users can clip the corresponding audio segment with the given *start* and *duration* attributes. Unlike other publicly available corpora that only provide normalized transcripts, Libriheavy includes richer information such as punctuation, casing, and text context. The text context is the transcription of the preceding utterances, located in the *pre_texts* entry, with a default length of 1000 bytes. There are also *begin_byte* and *end_byte* attributes, which allow users to easily slice any length of text context from the original book pointed to by the *text_path* attribute. Of course, there are other supplementary entries that might be usefull for other tasks, such as *id*, *speaker*, etc.

2.2.2. Evaluation Sets

As mentioned above, we have three evaluation sets in Libriheavy, namely dev, test-clean, test-other. We ensure that the evaluation sets have no overlapping speakers and books in the training set. To make the evaluation sets contain as many speakers and books as possible while not dropping out too much training data, we filtered out speakers and books with shorter durations as candidates. We then determine the *clean* speakers and *other* speakers using the same method as in [8] and divide the candidates into *clean* and *other* pool. We randomly select 20 hours of audio from the *clean* pool, half of which forms the test-clean set and the other half is appended to the dev set. We follow the same procedure for the *other* pool. Librilight ensures that audio files from the LibriSpeech evaluation sets are not present in the corpus, therefore, the LibriSpeech evaluation sets can also be used as our evaluation sets.

3. AUDIO ALIGNMENT

This section describes the creation pipeline of the Libriheavy corpus. The key task of audio alignment is to align the audio files to the corresponding text and split them into short segments, while also excluding segments of audio that do not correspond exactly with the aligned text. Our solution presented here is a general pipeline that can be applied to other data generation tasks as well. The implementation of all the following algorithms and corresponding scripts are publicly available[4].

3.1. Downloading text

To align the audio derived from audiobooks, we require the original text from which the speaker read the audiobook. From the metadata provided by Librilight, we can obtain the URL of the textbook for each audio file. We have written scripts to automatically extract the text and download the sources for all audiobooks. We then apply simple clean-up procedures such as removing redundant spaces and lines to the text sources.

3.2. First alignment stage

The goal of this stage is to locate the audio to its corresponding text segments (e.g. chapter) in the original book. First, we obtain the automatic transcript of the audio file. Then we treat the automatic transcript as **query** and the text in the original book as **target** [5], and find the close matches (Sec 3.2.2) for each elements in the query over the target. Finally, we determine the text segment of the audio by finding the longest increasing pairs (Sec 3.2.3) of query elements and their close matches. Note, we did not use the VAD tool provided by Librilight for audio segmenting and as our algorithm requires a relatively long text to guarantee its accuracy.
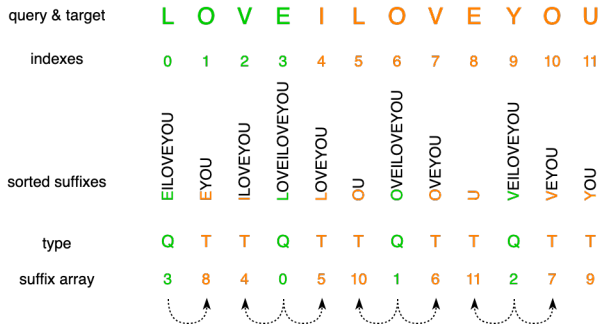
3.2.1. Transcribe audios

The audios in Librilight have a large variance in duration, from a few minutes to hours. To avoid excessive computation on long audio files, we first split the long audio into 30-second segments with 2 seconds of overlap at each side, and then recognize these segments with an ASR model trained on Librispeech. Finally, we combine the transcripts that belong to the same audio by leveraging the timestamps of the recognized words.

3.2.2. Close matches

Now we have the automatic transcript and the original book for each audio. To obtain the most similar text segment in the original book of the automatic transcript roughly, we propose the *close matches*. First, we concatenate query and target to a long sequence (target follows query), then a suffix array is constructed on the sequence using the algorithm in [16]. The *close matches* of the element in query position i is defined as two positions in the original sequence that are within the target portion, and which immediately follow and precede, in the suffix array, query position i. This means that the suffixes ending at those positions are reverse-lexicographically close to the suffix ending at position i. Figure 1 shows a simple

---

query & target: L O V E I L O V E Y O U

indexes: 0 1 2 3 4 5 6 7 8 9 10 11

sorted suffixes: EILOVEYOU | EYOU | ILOVEYOU | LOVEILOVEYOU | LOVEYOU | OU | OVEILOVEYOU | OVEYOU | U | VEILOVEYOU | VEYOU | YOU

type: Q T T Q T T Q T T Q T T

suffix array: 3 8 4 0 5 10 1 6 11 2 7 9

Fig. 1: Example of finding close matches for a query (*LOVE*) over the target (*ILOVEYOU*). The dash arrows point from the query elements to their close matches.

example of finding the close matches of query *"LOVE"* over target *"ILOVEYOU"*.

3.2.3. Longest increasing pairs

Let us think of those *close matches* which we obtained above as a set of (i, j) pairs, where i is an index into query sequence and j is an index into the target sequence. The query and its corresponding segment in the target should be monotonic aligned, so we can get the approximate alignment between the two sequences by finding the longest chain of pairs: $(i_1, j_1), (i_2, j_2), ... (i_N, j_N)$, such that $i_1 <= i_2 <= ... <= i_N$, and $j_1 <= j_2 <= ... <= j_N$.

3.3. Second alignment stage

From the longest chain obtained from the previous step, we can roughly locate the region in the target sequence relative to the query. At this stage, we use the Levenshtein alignment [17] to find the best single region of alignment between the recognized audio (query) and the text segment (obtained by the longest chain pairs). Since Levenshtein alignment is a quadratic time complexity algorithm, and will be very inefficient for long sequences. We can use the traceback through the pairs in the longest chain as the backbone for the Levenshtein alignment, so that we limit the Levenshtein alignment into blocks defined by the (i, j) positions in this traceback. By concatenating the Levenshtein alignments of all the blocks along the query index, we obtain the Levenshtein alignment of the whole query.

3.4. Audio segmentation

The goal of audio segmentation is to break long audio into shorter segments, ranging from 2 seconds to 30 seconds, which are more suitable for ASR training. We use a two-stage scoring method to search for good segmentations [6]. All books in LibriVox have punctuation, so we decided to split the sentence only at punctuation indicating the end of a sentence, namely, ".", "?" and "!" [7]. We select the positions

---

[6]The scores mentioned below will be normalized to the same scale, so none of the scores would dominate the final score.

[7]Our toolkit also supports splitting sentences at a certain threshold of silence.

in the alignment that follow chosen punctuations as Begin Of a Segment (BOS) and the positions followed by chosen punctuations as End Of a Segment (EOS), then we compute scores for these positions:

- The number of silence seconds this position follows or is followed by, up to 3 seconds.
- The score corresponding to the number of insertions, deletions and substitutions within a certain region of this position.

Each pair of BOS and EOS forms a segment. The following rule is applied to assign scores to potential segments:

- The score of BOS plus the score of EOS.
- A score related to the duration of the segment, which guarantees the duration is in the range of 2 to 30 seconds and encourages a duration between 5 to 20 seconds.
- A bonus for the number of matches in the alignment.
- A penalty for the number of errors in the alignment.

For each BOS, we find the 4 best-scoring EOS and vice versa. We then append the preceding 2 sets of segments to get a list of candidate segments. We determine the best segmentations by getting the highest-scoring set of segments that do not overlap. In practise, to avoid dropping out too much audio, we allow some kind of overlap if the overlapping length is less than a quarter of the segment.

4. EXPERIMENTS

In this section, we present the baseline systems and experimental results for two popular models, namely CTC-Attention [18] and neural transducer [2]. We then compare the performance between the models trained on normalized text and texts with punctuation and casing.

4.1. CTC-Attention baseline system

We build the CTC-Attention baseline using the Wenet [19] framework. We use the classic setup of Wenet toolkit which consists of a 12-layer Conformer [20] encoder and a 6-layer Transformer decoder. The embedding dimension is set to 512. The kernel size of the convolution layers is set to 31. The feedforward dimension is set to 2048. The modeling units are 500-class Byte Pair Encoding (BPE) [21] word pieces. The loss function is a logarithmic linear combination of the CTC loss (weight = 0.3) and attention loss with label smoothing (weight = 0.1). The input features are 80-channel Fbank extracted on 25 ms windows shifted by 10 ms with dither equal 0.1. SpecAugment [22] and on-the-fly Speed perturbation [23] are also applied to augment the training data. During training, we use the Adam optimizer [24] with the maximum learning rate of 0.002. We use the Noam [25] learning rate scheduler with 25k warm-up steps.

The model is trained for 90, 60 and 15 epochs on the small, medium and large subsets, respectively. Table 2 shows the Word Error Rate (WER) of the models on Libriheavy test

Table 2: The WERs of LibriSpeech (ls) and Libriheavy (lh) test sets on CTC-Attention system.

subset	ls-clean	ls-other	lh-clean	lh-other
small	5.76	15.60	6.94	15.17
medium	3.15	7.88	3.80	8.80
large	2.02	5.22	2.74	6.68

Table 3: The WERs of LibriSpeech (ls) and Libriheavy (lh) test sets on transducer system.

subset	ls-clean	ls-other	lh-clean	lh-other
small	4.05	9.89	4.68	10.01
medium	2.35	4.82	2.90	6.57
large	1.62	3.36	2.20	5.57

Table 4: The Libriheavy WERs and CERs on transducer system trained on texts with punctuation and casing.

subset	WER		CER	
	lh-clean	lh-other	lh-clean	lh-other
small	13.04	19.54	4.51	7.90
medium	9.84	13.39	3.02	5.10
large	7.76	11.32	2.41	4.22

Table 5: The comparison of WERs between models trained on Upper case No Punctuation (UNP) and Casing with Punctuation (C&P).

subset	text	ls-clean	ls-other	lh-clean	lh-other
small	UNP	4.05	9.89	4.68	10.01
	C&P	4.51	10.84	5.16	11.12
medium	UNP	2.35	4.82	2.90	6.57
	C&P	2.45	5.03	3.05	6.78
large	UNP	1.62	3.36	2.20	5.57
	C&P	1.72	3.52	2.28	5.68

sets. As a reference, we also show the WER on the LibriSpeech test sets. The N-Best hypotheses are first generated by the CTC branch and then rescored by the attention branch. Note that for the LibriSpeech results, we apply some simple text normalization, such as converting numbers to their corresponding text and converting abbreviations (e.g "Mr." to "Mister") on the hypotheses to make it compatible with the LibriSpeech transcripts. We also apply these normalization procedures in the following experiments.

4.2. Transducer baseline system

We build the transducer baseline system using the icefall [8] which is one of the projects in the Next-gen Kaldi toolkit. Icefall implements a transformer-like transducer system, which consists of a encoder and a stateless decoder [26]. Different from the setting in [26] which only has an embedding layer, an extra 1-D convolution layer with a kernel size of 2 is added on top of it. The encoder used in this baseline is a newly proposed model called Zipformer. The Zipformer paper has not been released yet, but the implementation details and training pipeline can be found in icefall [9]. We use the default setting in the Zipformer LibriSpeech recipe in icefall for all the following experiments.

The same as CTC-Attention baseline system, we train the model for 90, 60 and 15 epochs for the small, medium and large subsets, respectively. Table 3 shows the decoding results of the models trained on different training subsets, and the WERs on LibriSpeech and Libriheavy test sets are presented. We use the beam search method proposed in [27] which limits the maximum symbol per frame to one to accelerate the decoding.

4.3. Training with punctuation and casing

This section benchmarks the performance of models trained on texts with punctuation and casing, and compares them with the performance of models trained on normalized texts. The

system setting is almost the same as the transducer baseline system mentioned above. The only difference is that we adopt 756-class BPE word pieces rather than 500 for modeling, because we open the *fallback_bytes* flag when training the BPE model to handle rare characters, so we need an additional 256 positions for bytes. Table 4 shows the WERs and Char Error Rate (CER) of models trained on texts with punctuation and casing.

Table 5 compares the results of systems trained on normalized texts (upper case without punctuation) and unnormalized texts (casing with punctuation). In this experiment, we normalized both the transcripts and decoding results to upper case and removed the punctuation when calculating the WERs. From the results, the performance gap between two types of training texts is large when the training set is small, but as the training set grows, the gap becomes negligible. This indicates that when the training set is large enough, the style of training texts will not make much difference on performance, while training with texts with punctuation and casing brings us more information and flexibilities.

5. CONCLUSION

We release a large-scale (50,000 hours) corpus containing punctuation, casing and text context, which can be used in various of ASR tasks. We also propose and open-source a general and efficient audio alignment toolkit, which makes constructing speech corpora much easier. Finally, we conduct solid experiments on the released corpus, and the results show that our corpus is of high quality and demonstrates the effectiveness of our creation pipeline.

---

[8] https://github.com/k2-fsa/icefall
[9] https://github.com/k2-fsa/icefall/blob/master/egs/librispeech/ASR/zipformer/zipformer.py

6. REFERENCES

[1] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006.

[2] A. Mohamed A. Graves and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*, Vancouver, 2013.

[3] William Chan, Navdeep Jaitly, et al., "Listen, attend and spell," in *Proc. ICASSP*, Shanghai, 2016.

[4] George E Dahl, Dong Yu, Li Deng, and Alex Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE/ACM Trans. on Audio, Speech, and Language Processing*, 2011.

[5] Douglas B. Paul and Janet M. Baker, "The design for the Wall Street Journal-based CSR corpus," in *Speech and Natural Language: Proceedings of a Workshop Held at Harriman*, 1992.

[6] Godfrey, John J., and Edward Holliman., "Switchboard-1 release 2 ldc97s62.," 1993.

[7] Cieri, Christopher, et al., "Fisher english training speech part 1 transcripts ldc2004t19.," 2004.

[8] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *Proc. ICASSP*, Brisbane, 2015.

[9] Guoguo Chen, Shuzhou Chai, Guanbo Wang, Jiayu Du, Wei-Qiang Zhang, Chao Weng, Dan Su, Daniel Povey, et al., "Gigaspeech: An evolving, multi-domain asr corpus with 10,000 hours of transcribed audio," in *Proc. Interspeech*, Brno, 2021.

[10] Daniel Galvez, Greg Diamos, Juan Ciro, Juan Felipe Cerón, et al., "The people's speech: A large-scale diverse english speech recognition dataset for commercial usage," 2021.

[11] Vineel Pratap, Qiantong Xu, Anuroop Sriram, Gabriel Synnaeve, and Ronan Collobert, "MLS: A large-scale multilingual dataset for speech research," in *Proc. Interspeech*, 2020.

[12] Kai Wei, Thanh Tran, Feng-Ju Chang, et al., "Attentive contextual carryover for multi-turn end-to-end spoken language understanding," in *Proc. ASRU*, 2021.

[13] Shuo-Yiin Chang, Chao Zhang, Tara N Sainath, Bo Li, and Trevor Strohman, "Context-aware end-to-end ASR using self-attentive embedding and tensor fusion," in *Proc. ICASSP*, Rhodes, 2023.

[14] J. Kahn, M. Rivière, W. Zheng, E. Kharitonov, Q. Xu, P. E. Mazaré, J. Karadayi, V. Liptchinsky, R. Collobert, et al., "Libri-light: A benchmark for asr with limited or no supervision," in *Proc. ICASSP*, Barcelona, 2020.

[15] Piotr Żelasko, Daniel Povey, Jan "Yenda" Trmal, and Sanjeev Khudanpur, "Lhotse: a speech data representation library for the modern deep learning ecosystem," 2021.

[16] Juha Kärkkäinen, Peter Sanders, and Stefan Burkhardt, "Linear work suffix array construction," *Journal of the ACM*, 2006.

[17] Vladimir I Levenshtein et al., "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*. Soviet Union, 1966.

[18] Shinji Watanabe et al., "Hybrid CTC/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, 2017.

[19] Zhuoyuan Yao, Di Wu, Xiong Wang, Binbin Zhang, et al., "Wenet: Production oriented streaming and non-streaming end-to-end speech recognition toolkit," in *Proc. Interspeech*, 2021.

[20] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, et al., "Conformer: Convolution-augmented Transformer for Speech Recognition," in *Proc. Interspeech*, Shanghai, 2020.

[21] Rico Sennrich, Barry Haddow, and Alexandra Birch, "Neural machine translation of rare words with subword units," in *Proc. ACL*, Berlin, 2016.

[22] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, et al., "Specaugment: A simple data augmentation method for automatic speech recognition," in *Proc. Interspeech*, Graz, 2019.

[23] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur, "Audio augmentation for speech recognition," in *Proc. Interspeech*, Dresden, 2015.

[24] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Computer Science*, 2014.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, et al., "Attention is all you need," in *Proc. NIPS*, Long Beach, 2017.

[26] Mohammadreza Ghodsi, Xiaofeng Liu, James Apfel, et al., "RNN-transducer with stateless prediction network," in *Proc. ICASSP*, Barcelona, 2020.

[27] Wei Kang, Liyong Guo, Fangjun Kuang, Long Lin, Mingshuang Luo, Zengwei Yao, Xiaoyu Yang, Piotr Żelasko, and Daniel Povey, "Fast and parallel decoding for transducer," in *Proc. ICASSP*, Rhodes, 2023.