Development of a Low-Latency and Real-Time Automatic Speech Recognition System

Chee Siang Leow
University of Yamanashi
Kofu, Yamanashi, Japan
cheesiang_leow@alps-lab.org

Tomoaki Hayakawa
University of Yamanashi
Kofu, Yamanashi, Japan
kurotomo@alps-lab.org

Hiromitsu Nishizaki
University of Yamanashi
Kofu, Yamanashi, Japan
hnishi@yamanashi.ac.jp

Norihide Kitaoka
Toyohashi University of Technology
Toyohashi, Aichi, Japan
kitaoka@tut.jp

***Abstract*—In this study, a real-time automatic speech recognition (ASR) system based on the Kaldi ASR toolkit, with low-latency and customizable models, without any internet connection, was developed. The proposed ASR system includes a voice activity detection (VAD) module and an audio transmitter as a front-end speech processing and a decoder for the received audio signals. The ASR system was evaluated in terms of ASR accuracy and speech processing speed. Consequently, the ASR system achieved high ASR accuracy on the CSJ (Corpus of Spontaneous Japanese) test set with super low-latency.**

***Index Terms*—automatic speech recognition, low-latency, real-time, stand-alone**

I. Introduction

Recently, real-time automatic speech recognition (ASR) services have been used in various applications such as smart home devices and online conference tools. So far, real-time ASR services, such as Google Cloud Speech-to-text API (Google API) [1], IBM Watson Speech-to-text cloud services [2], and Rev.ai Speech-to-text API [3] are available in the cloud. Furthermore, we can easily use these ASR services on various applications through the provided API. These cloud services have achieved sufficiently high ASR accuracy and high-speed processing of input speech. Moreover, recent studies have focused mainly on end-to-end speech recognition such as ESPnet [4] which shows significant results on ASR system. However, it is well known that end-to-end system requires more data than a deep neural network (DNN)-hidden Markov model (HMM)-based ASR system to achieve high performances of ASR recognition.

Kimura et al. [5] compared the ASR performance of Japanese speech using Google API and "Kaldi" [6], an ASR toolkit commonly used globally. This study confirmed that the Kaldi system achieved high accuracy of speech recognition when the training environment for acoustic and language models was closed to the evaluation environment, while the Google API could achieved certain high recognition accuracy and processing speed in various environments. Although Google API was superior from a general-purpose perspective, the experiments indicated that the Kaldi is better than Google API at domain-adapted acoustic and language models, such as in a spoken dialog system for a particular domain.

In this study, we reported the development and evaluation of a stand-alone, low-latency, and relatively high-accuracy real-time ASR system for Japanese based on the Kaldi toolkit. We compared our ASR system with the widely used Google API. Our ASR system has an advantage over Google API in terms of the ability to customize acoustic and language models and a recognition dictionary. In the experiment, we evaluated our ASR system in terms of ASR accuracy and speed compared with Google API on the in-domain speech recognition task. Consequently, while our system's ASR accuracy was exceeded that of Google API on the CSJ test set, its speed (latency) also exceeded that of the Google API, which requires an internet connection.

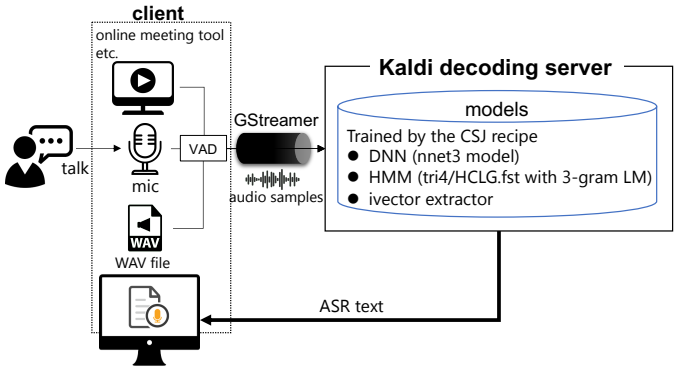

Fig. 1. System architecture of the real-time automatic speech recognition system.

II. Real-time ASR System

A. System Architecture

Figure 1 shows the system architecture of the developed ASR system. The ASR system can directly recognize voice inputted from a microphone in real-time, and it also has functions for accepting direct input of computer audio using file input and a virtual audio input/output device. The virtual audio input/output allows us to realize real-time ASR for meeting voices outputted from various online meeting and video streaming tools, such as Zoom[1] and YouTube. An inputted voice is processed through VAD, and only utterance parts are transmitted to an ASR decoder via GStreamer framework [7].

The architecture of our ASR system is a server-client model, but the recognition server (decoder) can work either on a computer on the Internet or a local computer. The client (front-end) module and decoder can run on Linux, macOS, and Windows Subsystem for Linux in Windows10; therefore, it can work as a stand-alone system on most computers.

[1]https://zoom.us/

978-1-7281-9802-6/20/$31.00 ©2020 IEEE

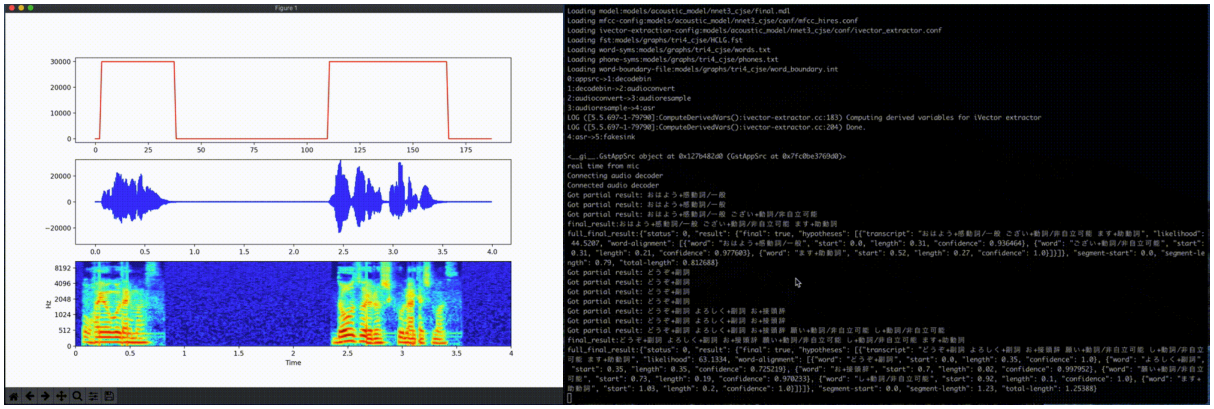

Fig. 2. Real-time VAD and spectrogram visualization.

B. Decoder

The GStreamer plugin, gst-kaldi-nnet2-online [8], was used as a decoder. The detected voice is sent from the client to the GStreamer, and the voice is sequentially converted to a sampling frequency of 16 kHz and input to the decoder. When the VAD at the client detects the end of an utterance, it sends an end-of-stream signal to the GStreamer. Then, the decoding is completed, and the final 1-best result is determined. At this point, the GStreamer returns the information such as recognition time, confidence level for each word, and the length of audio that being recognized in JSON format to the client, and the transcribed texts are displayed on the screen. Our system did not stop on recording and decoding after the first result been displayed; for the consideration of usage such as lectures or conferences where the speeches can be long, our system displayed the previous 2 results and the latest result. The method for displaying the ASR result can be easily changed from the front-end code, for example, by stacking all results in a single array to be displayed.

C. VAD

The Python binding of the WebRTC VAD [2] was used in our ASR system. Our ASR system records the audio by 1024 chunks of the selected sampling rate. The VAD uses the last 20ms speech frame from the recorded chunks of audio to compute a result of silence or speech. We implemented the VAD on client-side for the flexibility of various client-side implementations such as usage on unity. The VAD is used to determine whether to send the recorded chunks of audio; if silence is detected, the client-side sends silence clip of audio instead of recorded audio. If the server-side received a silent clip, the GStreamer decoder will be silent and will not decode the specific audio chunks. This process is used to reduce the number of background noise to be decoded as filler.

We introduce a $timeout$ parameter to count the number of time of VAD being detected silent. The $timeout$ parameter can be used to tune how fast to finalize the decoder that is causing performance difference of ASR results. In this study, we set the $timeout$ as 5, meaning if the server-side detected 5 chunks of silent audio, server-side will send the signal of EOS to the decoder to finalize the ASR result and the $timeout$ will be reset to 0. Finally, for an option, we provide a matplotlib[3] based real-time VAD and spectrogram visualization to analyze the recorded speeches (Fig.2), which is useful to analyze the recorded speech and ASR result.

D. ASR Models and Dictionary

The acoustic model is a hybrid model comprising the HMM and DNN. The model was trained with all speeches of CSJ [9] except for the dialog speeches, Japanese Newspaper Article Speech Database (JNAS) [10], Senior JNAS (S-JNAS) [11], and Elderly Adults Read Speech Corpus [12]. The total duration of speech data for acoustic modeling is approximately 750 h. The acoustic model was trained following the CSJ recipe in the Kaldi toolkit. Finally, we obtained the nnet3 (TDNN, Time Deray Neural Network) [13] model and the HMM-based WFST with the 3-gram language model.

The word 3-gram-based language model was trained from a transcription of all the CSJ lectures and speeches and the Balanced Corpus of Contemporary Written Japanese (BCCWJ) [14]. Text sentences in these corpora are word-segmented by a morphological analyzer MeCab with a UniDic-2.3.0 [15] dictionary before making a recognition dictionary and training a language model. The number of words entered in the dictionary was about 164k.

III. Experiments

A. Evaluation Measures

We evaluated our speech recognition system based on the following points:

- ASR accuracy (Word Error Rate, WER) on the CSJ eval3 test set,
- Real-Time Factor (RTF) and latency, and
- Effect of different microphones on WER.

B. Experimental Conditions

We evaluated our ASR system using the CSJ eval3 test set. The ASR of the test set was performed from the speech files and evaluated by WER. We also evaluated Google API for the same test set for comparison. We measured our ASR

[2]https://github.com/wiseman/py-webrtcvad

[3]https://matplotlib.org/

978-1-7281-9802-6/20/$31.00 ©2020 IEEE

system's latency by taking the difference between the time of the end of utterances received and time that the recognition result was confirmed. Since we do not know how the recorded speeches are being processed by Google API, we cannot compare the latency of the Google API ASR system with our ASR system. We measured the RTF by measuring the length of ASR recognition time taken from sending an audio file to Google API cloud service to the time of the ASR result received from Google API. Our ASR system RTF is measured under the same condition with the Google API.

RTF is the ratio of the ASR processing time to the length of time of the utterance audio clip y to be recognized. For the comparison of RTF, we used a 2017 MacBook Pro (MBP) for both ASR systems with CSJ eval3 test set. The RTF calculation evaluated using the following Equation 1.

$$RTF = \frac{time(ASR(y))}{length(y)} \tag{1}$$

Four type of microphones were prepared: an omnidirectional Tabletop Microphone, 2017 MacBook Pro (MBP) built-in mic., AirPods Pro from Apple Inc. (earphones with a microphone), and Respeaker Mic Array v2.0 (4-microphones array). Five men uttered four speeches selected from academic lectures in CSJ, and each speech was recorded to a WAVE file by these microphones. The recorded speeches were speech-recognized by our ASR system and Google API. We investigated how the type of microphone affected WER.

C. Normalization of Reconized Words

Since the representation of the recognized words of our ASR system different from Google API, we normalized both the ASR system recognized words with the following procedures before calculating the WERs.

1) Convert alphabets to lowercase
2) Convert full-width characters to half-width characters
3) Convert digits number to kanji number
4) Remove sp symbols
5) Remove fillers
6) Remove interjection words

D. Results and Discussions

Table I shows the WERs and RTF (latency) results for the CSJ eval3 test set. Table I indicates that our ASR system exceeded Google API on both the WER and RTF. This is partly due to the closed environment of acoustic and language models. However, this result suggests that customized models and recognition dictionaries significantly improve recognition accuracy. Besides, in terms of recognition speed, RTF surpasses to the Google API because our ASR system can work in a stand-alone operation, meaning that it does not require internet connection. The recognition result was confirmed within 0.5 seconds after the end of the utterance. It seems to work much faster than the Google API sensuously.

Table II also shows WERs of utterances recorded by five adult men using four sorts of microphones. The Google API produced stable WERs even when the microphone changes; however, our ASR system achieved very different WERs due to changes in the microphone environment. Google's model was trained with audio recordings in various environments, making the model more robust against changes in the environment. Therefore, we also need to train an acoustic model using audio recorded in various environments.

TABLE I
WERs AND RTF (LATENCY) EVALUTIONS ON THE CSJ EVAL3 TEST SET.

	Google API	Our ASR
WER [%]	28.16	6.45
RTF	0.324	0.167
Latency [s]	—	0.436

TABLE II
WERs FOR THE FIVE SPEAKERS' UTTERANCES RECORDED BY FOUR TYPES OF MICROPHONES.

Mic. type	Google API	Our ASR
Tabletop mic.	17.65	10.78
MBP built-in mic.	11.76	9.80
AirPods Pro	14.71	19.61
Respeaker	12.75	15.69
average	14.22	13.97

IV. Conclusions

In this study, we presented and evaluated a low-latency real-time ASR system and compared it with Google API. Our system works with lower-latency than Google API, and the recognition accuracy exceeded that of Google API in a closed environment. For an option, we provided a matplotlib based visualization on the analyzed spectrogram and VAD. Because the extra computation could occur on visualization, the ASR latency might be affected. The speed of visualization could be improved by changing from matplotlib to multithread PyQT5[4]-based visualization. We expect to add more analyzers such as the speech audio's jitter, F0 frequency, shimmer, average word-level amplitude visualization in the future. This can help us understand more about the problems that affect the ASR performance in real-world environments.

In the future, we plan to improve the front end of the ASR system so that it can withstand noisy environments and improve the acoustic model to make it more practical to use. Furthermore, we expect to compare the latency with more ASR systems.

Acknowledgment

This work was supported by JSPS KAKENHI Grant-in-Aid for Scientific Research (A) Grant Number 19H01125. Moreover, a part of this work was also supported by Hoso Bunka Foundation.

References

[1] Google: Speech To Text API https://cloud.google.com/speech-to-text, Referred on 7/August/2020.
[2] IBM: Watson Speech To Text API https://www.ibm.com/cloud/watson-speech-to-text, Referred on 7/August/2020.
[3] Rev.ai: Speech To Text API https://www.rev.ai/, Referred on 7/August/2020.

[4]https://www.riverbankcomputing.com/software/pyqt/

978-1-7281-9802-6/20/$31.00 ©2020 IEEE

[4] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, T. Ochiai "ESPnet: End-to-End Speech Processing Toolkit," Proc. of INTERSPEECH2018, 2018, pp.2207-2211.
[5] T. Kumura, T. Nose, S. Hirooka, A. Ito, "Comparison of Speech Recognition Performance Between Kaldi and Google Cloud Speech API," Proc. of IIH-MSP 2018, 2018, pp.109-115.
[6] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi Speech Recognition Toolkit," Proc. of ASRU 2011, 2011.
[7] GStreamer: Open Source Multimedia Framework, https://gstreamer.freedesktop.org/, Referred on 29/June/2020.
[8] T. Alumäe, "Full-duplex Speech-to-text System for Estonian," Human Language Technologies - The Baltic Perspective, vol.268, 2014, pp.3-10.
[9] K. Maekawa, "Corpus of Spontaneous Japanese: Its design and evaluation," Proc. of the ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition (SSPR 2003), pp. 7–12, 2003.
[10] K. Itou, M. Yamamoto, K. Takeda, T. Matsuoka, T. Kobayashi, K. Shikano, and S. Itahashi, "JNAS: Japanese speech corpus for large vocabulary continuous speech recognition research",Journal of the Acoustical Society of Japan 1999,volume 20,pp.199–206,
[11] S. Kiyohiro, Laboratries of Image Information Science and Technology, Nara Institute of Science and Technology, Speech Resources Consortium, "Japanese Newspaper Article Sentences Read Speech Corpus of the Aged (S-JNAS)" http://research.nii.ac.jp/src/en/S-JNAS.html, Referred on 7/August/2020.
[12] M. Fukuda, R. Nishimura, H. Nishizaki, Y. Iribe, N. Kitaoka, "A New Corpus of Elderly Japanese Speech for Acoustic Modeling, and a Preliminary Investigation of Dialect-Dependent Speech Recognition," Proc. of O-COCOSDA, 2019, pp.1–6.
[13] V. Peddinti, D. Povey, S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," Proc. of INTERSPEECH2015, 2015, pp.3214-3218.
[14] Center for Corpus Development NINJAL, "The Balanced Corpus of Contemporary Written Japanese (BCCWJ)" https://pj.ninjal.ac.jp/corpus_center/bccwj/en/, Reffered on 7/August/2020.
[15] Y. Den, J. Nakamura, T. Ogiso, H. Ogura. "A Proper Approach to Japanese Morphological Analysis: Dictionary, Model, and Evaluation", Proc. of the sixth international conference on Language Resources and Evaluation (LREC 2008), 2008, pp.1019-1024.

978-1-7281-9802-6/20/$31.00