Customization of IBM Intu's Voice by Connecting Text-to-Speech Services with a Voice Conversion Network

Jongyoon Song
ECE, Seoul National University
coms1580@snu.ac.kr

Jaekoo Lee
ECE, Seoul National University
jaekoolee@snu.ac.kr

Hyunjae Kim
ECE, Seoul National University
preskim@snu.ac.kr

Euishin Choi
Client Innovation Lab, IBM Korea
choies@kr.ibm.com

Minseok Kim
Developer Outreach Team, IBM Korea
misekim@kr.ibm.com

Sungroh Yoon*
ECE, Seoul National University
sryoon@snu.ac.kr

Abstract

IBM has recently launched Project Intu, which extends the existing web-based cognitive service Watson with the Internet of Things to provide an intelligent personal assistant service. We propose a voice customization service that allows a user to directly customize the voice of Intu. The method for voice customization is based on IBM Watson's text-to-speech service and voice conversion model. A user can train the voice conversion model by providing a minimum of approximately 100 speech samples in the preferred voice (target voice). The output voice of Intu (source voice) is then converted into the target voice. Furthermore, the user does not need to offer parallel data for the target voice since the transcriptions of the source speech and target speech are the same. We also suggest methods to maximize the efficiency of voice conversion and determine the proper amount of target speech based on several experiments. When we measured the elapsed time for each process, we observed that feature extraction accounts for 59.7% of voice conversion time, which implies that fixing inefficiencies in feature extraction should be prioritized. We used the mel-cepstral distortion between the target speech and reconstructed speech as an index for conversion accuracy and found that, when the number of target speech samples for training is less than 100, the general performance of the model degrades.

1. Introduction

One of the major interests for artificial intelligence (AI) researchers is to improve the naturalness of communication and interaction with machines. Recently, machine learning approaches using neural networks have achieved outstanding performance in various AI applications and some have already been commercialized by several enterprises [1–3].

One of the most significant AI applications is the Internet of Things (IoT). It is natural to utilize AI technology in IoT because IoT demands dynamic data communication between devices and users in order to provide flexible and integrated services. Amazon Alexa [1], Google Home [2], and IBM Project Intu [3] are examples of current intelligent personal assistant services.

An intelligent personal assistant service is a system that interacts with a user through text or voice. The main objective of the intelligent personal assistant service is to provide human language interactions to the user, similar to a human assistant.

We propose a method to improve the quality of user experiences with intelligent personal assistant services. Current commercial assistant services support preset voices only. Therefore, we implemented a service that allows users to customize the voice of the IBM Intu intelligent personal assistant. We expect users to have more rich and intimate experience by using a voice that they prefer.

IBM Watson [4] is an integrated cognitive service that provides partial AI services, such as text-based conversation [5], translation [6], etc. IBM Intu is middleware that provides an intelligent personal assistant service by leveraging the appropriate Watson services. Because Intu shares the data yielded by Watson services internally, it can perform a wider range of functions.

Developers can design Intu's process routine by creating and allocating modules for specific tasks.

*Corresponding author.

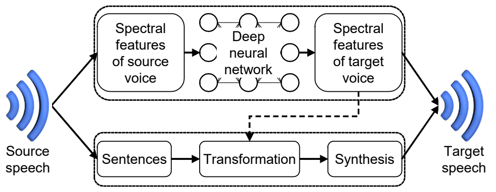

Figure 1. Mechanism for voice conversion using a neural network.

Additionally, developers may incorporate desired Watson services into each Intu module. Intu's modular structure allows us to perform experiments on it while using simplified structures.

We implemented a voice customization service by integrating Watson's text-to-speech (TTS) service [7], which generates speech from text sentences, and a voice conversion network (VCN). Voice conversion is the transformation of some source speech into the speech of a target voice, as depicted in Figure 1. The reason for including the TTS module in the voice customization service is that Intu first obtains the text corresponding to the content of the output speech and then generates the speech by using the TTS module.

A voice customization service can be achieved in two ways. One is to use a TTS model to directly generate the voice a user wants.

For example, DeepMind's WaveNet trains a model for multi-speaker identity and TTS mapping by exploiting speech data from multiple speakers [8]. However, for in a real world voice customization service environment, an increasing number of speaker identities should be supported. Therefore, a single integrated model with fixed complexity must be verified for the scalability of speaker identities.

The other way to customize a voice is to use a VCN to convert the output voice of a TTS model. The model we implemented uses a training module for speaker identity and a TTS module separately. Therefore, when a new speaker identity needs to be added to the model, we only need to train a new voice conversion module.

Most recent voice conversion models require parallel data for a target voice during training [9–12]. In other words, the speech data for two or more speakers pronouncing the same sentence is necessary. However, it is difficult for a real user to prepare parallel data. Additionally, the time frame for the parallel data must be aligned in order to train the frame-wise mapping model.

However, the model proposed in [13] does not require parallel data for the target voice during training. Furthermore, the input data and their labels are generated from same target speech, meaning that additional data alignment is unnecessary. Therefore, we decided to implement the voice conversion model proposed in [13] and integrate it with Intu.

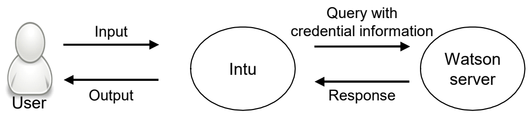

Figure 2. Data flow of the Intu service.

In this study, we implemented the simplified Intu-VCN model, which returns user speech in a customized voice. The model has been specialized for our experiments. We found ways to reduce the additional time required by the VCN and determined the appropriate number of target speech samples. The main contributions of our work are as follows:

1) We implemented a prototype of a voice customization service for the IBM Intu, intelligent personal assistant service, through a combination of Intu and a VCN.

2) We quantitatively investigated the problems of the time consumption and the number of target speech samples problems by conducting experiments in realistic situations.

The remainder of this paper is organized as follows: In Section 2, we briefly discuss related work on IBM Watson, Project Intu, TTS, and voice conversion. Preliminary knowledge that is required for better understanding of the proposed method is in Section 3. Section 4 describes the integrated model for Intu and a VCN in detail. In Section 5, we design and perform several experiments that were necessary for launching our voice customization model. Our conclusions are presented in Section 6.

2. Related work

2.1. IBM Watson and Project Intu

The IBM Watson service is an integrated cognitive system that provides the module-based functions of an AI service [4]. Additionally, Watson is designed with extensibility to allow users to assemble Watson instances of each service for additional applications.

If a user's device provides credential information for authentication and a query to Watson's server, the server verifies the credentials and returns a service response. For instance, the model we implemented sends a text-format sentence to the server and receives the corresponding speech from the Watson TTS service [7].

Project Intu is middleware that expands Watson into the IoT environment [3]. When an Intu device detects user input, the Watson service is used to provide the corresponding output to the device [3]. As shown in Figure 2, Intu sends queries, including credential information, to the server when using the Watson service. The main modules that comprise Intu are the extractor, blackboard, agent, and gesture.

The extractor is placed at the beginning of Intu's data processing module and preprocesses inputs so that the data are suitable for the following modules. For instance, the extractor converts the visual information from a camera into an image format or the voice information from a microphone into a text format.

In the blackboard, the input and output data of the IoT devices, as well as the intermediate data generated during processing, are shared between other modules. When agent and gesture subscribe to a specific type of data in the blackboard, they read and handle data in real time whenever new data are registered in the blackboard. Even if the data are in the same format, the blackboard redefines the data in a wrapped type to distinguish data during processing. For example, text obtained from input speech and a text response from a Watson service have different types.

The agent is a module that performs the sub-processes associated with assistant tasks. Agents internally invoke Watson services in general and execute tasks by combining sub-processes. In most cases, agents read data from the blackboard and send them to the Watson service in the form of a query in order to obtain an output, which is then passed back to the blackboard. The output of an agent is passed to successive agents as input or output by gestures. Examples of agents are the weather agent, which returns information about the weather, and the emotion agent, which classifies person's emotions as positive or negative.

The gesture is a module that handles Intu's output and sometimes calls a specific Watson service in the process. The gesture module performs actions such as outputting speech through a speaker or playing sounds. Additionally, actions such as sending a message to another device or controlling a device are also outputs of gestures. In other words, while agents are responsible for computing or processing data, gestures perform the passive role of transferring data to an output device or carrying out instructions contained in the data.

2.2. Text-to-Speech (TTS)

TTS is a task that generates audible speech from text. Typically, TTS methods can be divided into two types based on how the speech is synthesized [8, 14]. One is to find the optimal sequence of acoustic units from a database and concatenate them into speech [8,15,16]. In this method, large amounts of speech data are segmented into phonetic units and stored in the database [15, 16]. The other method is to generate waves directly by predicting acoustic features from the phonetic units of the parsed input text [17].

Both methods have advantages and disadvantages, and both are currently in use. Speech generated using the former method has high naturalness because the acoustic units are derived from real speech. However, sufficient numbers of acoustic units must be stored in order to create an optimal sequence for all text. The latter method only needs to store the model's parameters. However, speech reconstruction with parameters of acoustic features with a vocoder is vulnerable to loss of detailed features.

The unit selection technique, an example of the former method types, was firstly introduced in the past. Next, methods in which speech is directly reconstructed from the parameters of acoustic features, which are derived from hidden Markov models (HMMs), were proposed as representatives of the latter method type [14, 17].

Recently, there have been studies on applying dilated convolution or recurrent neural network (RNN) models to both speech synthesis method types in order to learn the context dependencies of speech [8, 18]. DeepMind's WaveNet [8] uses a dilated convolution model as its main component. Baidu's Deep Voice uses gated recurrent units to predict phoneme duration and F0 values, and synthesizes audio by using a bidirectional quasi-RNN layer, which is one of many RNN variants [18]. The Watson TTS model uses a deep bidirectional long short-term memory (DBLSTM) model to extract prosodic features from text [7, 19, 20].

The Watson TTS model concatenates optimal acoustic units into the output speech. Because the model has to store a large number of acoustic units for each speaker [19], it is difficult to add a new speaker identity with the small amount of data that a typical user can prepare.

The other TTS models discussed above also require large amounts of training data to learn the relationships between the linguistic features of text and the acoustic features of speech [8, 18–20]. For example, WaveNet requires 24.6 h of speech data for a single speaker TTS model [8] and Deep Voice requires approximately 20 h of speech data [18]. Watson's TTS model also requires hours of single speaker speech data to train its prosody target model [19, 20].

WaveNet is powerful because it can learn TTS tasks and multiple speaker identities in the same model. Furthermore, it reduces the required amount of speech data from each speaker because the internal representation of the model is shared between speakers [8]. However, because the two types of characteristics are learned in a single model, the process of learning speaker identities cannot be done separately. This means that the model must be able to learn an increasing number of speaker identities in the fixed model. As a result, we must confirm that the model is stable in terms of scalability of speaker identities.

In conclusion, we determined that replacing the Watson TTS model with another TTS model is meaningless. Therefore, we kept the Watson TTS model for the TTS portion of our voice customization model.

2.3. Voice conversion

In the past, one of the main methods of voice conversion was to map source speech onto target speech discretely [21,22]. In [21], codebooks for source voices, target voices, and mapping are constructed for voice conversion. In [22], the index of the target voice's codebook is mapped from the feature vector of the source speech by using a feedforward neural network.

Studies on preserving the continuity of acoustic features followed these early works. Most studies modeled the acoustic features of the source and target voices by using a Gaussian mixture model (GMM) or joint density GMM (JDGMM) [23–26]. However, these models could not prevent the loss of acoustic features, such as the features of time dependency, and displayed over-smoothing effects. There have been numerous studies aimed at solving these problems [26–29].

Various neural networks have been used in place of GMMs or JDGMMs for feature space modeling [9–13,30]. For instance, a restricted Boltzmann machine (RBM) and a deep belief network (DBN) have both been used to model the spectral distributions of the source and target voices [9, 10, 30]. Recently, there have been efforts to preserve the time dependency of speech by introducing long short-term memory (LSTM) and bidirectional LSTM [11, 13].

However, most recent research has been restricted to the use of parallel data, meaning the pronunciations of the source and target voices are paired for the same sentences [9–12]. Parallel data makes modeling difficult due to the expense of data collection and incomplete frame alignment [13].

In [13], the authors solved the problems associated with parallel data by extracting speaker-independent linguistic features. Because the model can extract both speaker-independent linguistic features and acoustic features from the target speech, the model does not require parallel data and avoids the frame alignment issue [13]. Furthermore, the model exploits DBLSTM for the robust learning of time dependencies [13]. Therefore, we adopted the model proposed in [13] to implement our voice customization service for Intu.

3. Preliminaries

3.1. Phonetic class posterior probabilities (PPPs)

Phonetic class posterior probabilities (PPPs) are a way to represent the linguistic state in each time frame of speech. The phonetic class can be set differently based on the scope of speech segmentation, such as for phonemes, triphones, or senones. The transcription of speech can be divided into phonemes. However, a phoneme is voiced differently depending on the surrounding context because of the phonemic rules and mechanics involved in speech production.

Triphones can be used to reduce this ambiguity. A triphone is a subdivision of phonemes based on the preceding and subsequent phonemes. However, the number of different triphones is very large (over 50,000). A senone reduces the number of possibilities by clustering groups of triphones with similar pronunciation patterns. Therefore, we adopted the senone for our phonetic class.

3.2. Mel-frequency cepstral coefficients (MFCCs)

Mel-frequency cepstral coefficients (MFCCs) are spectral representations of speech. The reason for using MFCC features for voice recognition is that the domain of mel-scale frequency reflects human sound perception. The higher a sound's frequency, the more difficult it is for humans to perceive a change in its frequency. The mel-scale frequency domain is a log-scaled domain that levelizes the human sound perception space.

3.3. Mel-cepstral coefficients (MCEPs)

Mel-cepstral coefficients (MCEPs) correspond to the coefficients when the spectrum of a signal is represented by an M-th order expression. This is shown in Eq. (1), which uses mel-cepstral analysis [31].

$$H(z) = \exp \sum_{m=0}^{M} c_\alpha(m)\tilde{z}^{-m} \tag{1}$$

$$\tilde{z}^{-1} = \frac{z^{-1} - \alpha}{1 - \alpha z^{-1}} \tag{2}$$

where $c_\alpha(m)$ is the m-th order of mel-cepstral coefficients of the spectrum $H(z)$ and the α is a constant for the first order all-pass function shown in Eq. (2) [31]. We used the MCEP feature as a representation of the spectral envelope in order to reconstruct the spectrum.

3.4. Feature-based maximum likelihood linear regression (fMLLR)

Speaker adaptation in speech recognition refers to the transformation process that is used to recognize the speech of a new speaker who was not used during the model's training. Feature-based maximum likelihood linear regression (fMLLR) is the process of performing speaker adaptation by transforming the features of speech so that a trained speech recognition model can be applied to the new speaker [32]. In order to accomplish this, a transformation matrix W is derived such that the likelihood of a new speaker's transformed speech data is maximized. The transformation matrix W is expressed as a concatenation of matrix A and a bias term b, as shown in Eq. (3).

$$W = [\,A \quad b\,] \tag{3}$$

The feature vector x for the speech is processed into ξ, as shown in Eq. (4). Finally, the transformed feature vector $\hat{x}$ is obtained using Eq. (5) [32].

$$\xi = \begin{bmatrix} x \\ 1 \end{bmatrix} \tag{4}$$

$$\hat{x} = W \cdot \xi \tag{5}$$

We transform the MFCC features of speech into fMLLR features within the speaker-independent auto speech recognition (SI-ASR) module in order to extract normalized features for the speaker.

3.5. Fundamental frequency (F0) and aperiodicity component (AC)

The fundamental frequency (F0) corresponds to the lowest frequency among the frequencies with periodicity in speech. When a voice is synthesized with a converted F0, the pitch of the voice is also converted. The aperiodicity component (AC) is a component that does not display periodic properties when the signal is analyzed in the frequency domain. This component includes the complex properties of the voice.

3.6. Deep bidirectional long short-term memory (DBLSTM)

In a basic RNN cell, the multiplication operation is repeated as the time step increases. Therefore, vanishing gradient or exploding gradient effects may occur when backpropagation through time is conducted. LSTM, one of the RNN model variants, does not exhibit the repetitive effects of multiplication because the cell state and the forget gate at the current time step are element-wise multiplied and the cell state is transmitted to the next time step.

Because the vanishing gradient and exploding gradient effects do not occur in LSTM, it is more suitable for learning long temporal dependencies. In general, because a single utterance can be composed of thousands of frames, it is appropriate to use LSTM, rather than a basic RNN, for speech analysis.

Bidirectional LSTM is a model that uses the hidden states of both forward and backward directional LSTM in order to obtain features at the current time step. A bidirectional LSTM model is more suitable for speech analysis than a unidirectional LSTM model because the acoustic features of speech are affected by bidirectional context.

For the output y_t at time step t, the relationship

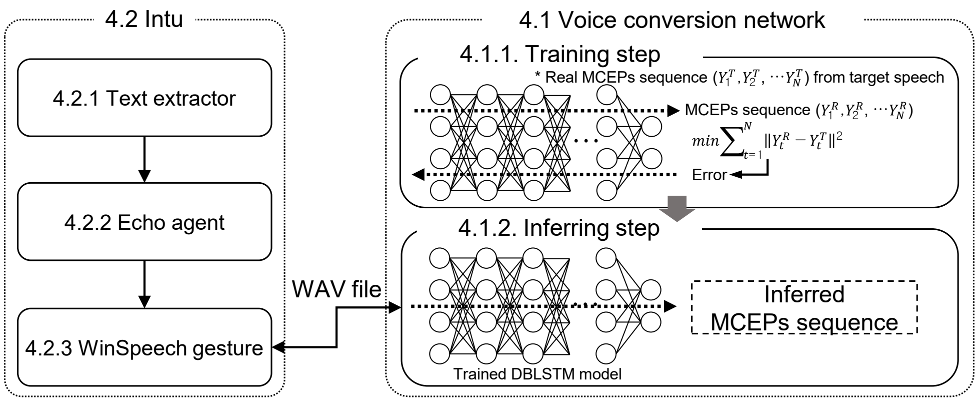
4.2 Intu
4.2.1 Text extractor
4.2.2 Echo agent
4.2.3 WinSpeech gesture
WAV file
4.1 Voice conversion network
4.1.1. Training step
* Real MCEPs sequence ($Y_1^T, Y_2^T, \cdots Y_N^T$) from target speech
MCEPs sequence ($Y_1^R, Y_2^R, \cdots Y_N^R$)
$min \sum_{t=1}^{N} \|Y_t^R - Y_t^T\|^2$
Error
4.1.2. Inferring step
Inferred MCEPs sequence
Trained DBLSTM model

Figure 3. Overall structure of the voice customization model.

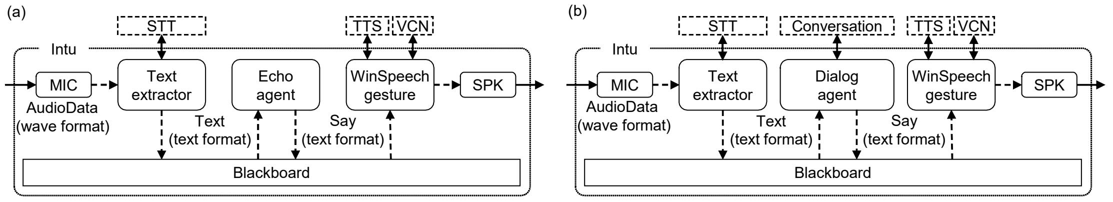

Figure 4. Echoing (a) and dialog (b) models with voice customization. Dashed box denotes components that exist in outer server. STT, conversation, and TTS components reside in the Watson server and VCN resides in the VCN server.

between h_t^f and h_t^b, which are the hidden states of forward and backward directional LSTM, respectively, satisfies:

$$y_t = W_o^f h_t^f + W_o^b h_t^b + b_o \tag{6}$$

where W_o^f is a linear transformation matrix for the hidden state of forward directional LSTM and W_o^b is the same for backward directional LSTM. b_o is a bias term.

DBLSTM is a structure that improves the complexity of the model by stacking bidirectional LSTM into several layers. At time step t, the input of the current layer is the sum of the hidden states of the forward and backward directional LSTM on the previous layer.

4. Implementation details

The overall structure of our model is presented in Figure 3. We employed the echoing model, depicted in Figure 4(a), for our experiments, using the minimum number of modules required for an Intu voice output process. In the model, when a user inputs speech data through Intu's microphone, the speech data are converted into text data by Watson's speech-to-text (STT) service [33]. The text data are then converted back into speech data in Intu's voice by Watson's TTS service [7]. Finally, the speech data are converted into the target speech by the VCN and output through a speaker. The actual conversation model can be implemented by replacing the echo agent with a dialog agent, as depicted in Figure 4(b).

Because we focus on issues in the VCN when it is adopted by Intu, we naïvely implemented VCN in an external server and transferred speech data through a simple file transfer protocol. When Intu communicates with the VCN server to substitute output speech data, Intu first stores the output speech data as an WAV file. Next, Intu sends the file to the designated directory of the VCN server. The VCN server then converts the input speech file and returns it to Intu. Finally, Intu outputs the converted speech through its speaker.

4.1. Voice conversion network

In this section, we discuss the process of training SI-ASR in the development stage, the process of training DBLSTM when it has received a sample of target speech data from a user, and the process by which the VCN converts Intu's speech. The mechanism we use is the same as that proposed in [13].

The VCN model consists of three stages, as shown

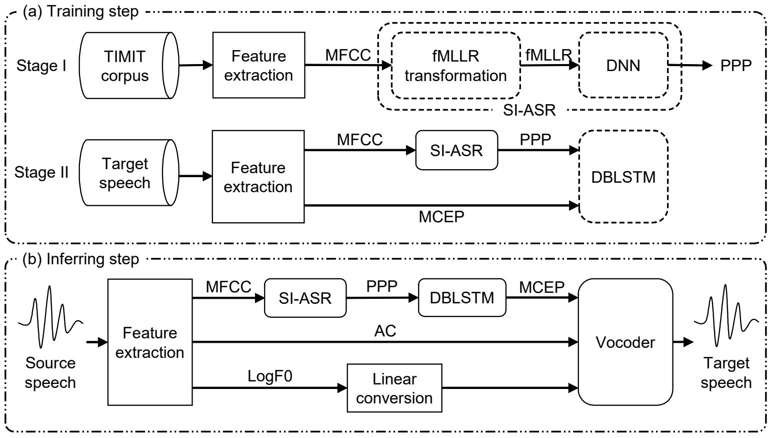

Figure 5. Description of voice conversion network [13]. Dashed components are to be trained at that stage.

in Figure 5. In stage 1, SI-ASR, which maps MFCCs of speech to PPPs, is trained using the TIMIT [34] corpus. In stage 2, DBLSTM, which maps PPPs to MCEPs of the target speech, is trained using the target speech sample. After training, the VCN uses the models learned in the previous stages to convert speech into a new voice, as shown in the inferring step.

All the input speech data in Figure 5 are sampled at a rate of 16kHz in a mono channel. For speech processing, we chose a window length of 25 ms and an overlapping length of 5 ms [13].

4.1.1. Training step.

SI-ASR, which is trained in stage 1, extracts PPP features. First, Kaldi [35], a speech recognition toolkit, extracts the MFCC features from the TIMIT corpus. Next, the fMLLR transformation model and DNN are trained in order. The structure of the DNN is a four-layered feedforward neural network in which the unit size of each hidden layer is 1024 and the output is calculated by an additional softmax layer. The dimensions of the MFCCs, fMLLR features, and PPPs are 13, 40, and 134, respectively. In stage 2, DBLSTM is trained using the PPPs from SI-ASR as input and the MCEPs as labels. DBLSTM consists of four layers of bidirectional LSTM. The MCEP features are extracted by SPTK [31], which is a speech analysis toolkit. The number of the units in the input layer is 134 and the numbers of the hidden units in the four bidirectional LSTM layers are 64, 64, 64, and 39, respectively. We used the Adam optimizer algorithm with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e - 08$. The cost function is the summation of the L2-norms at all time steps.

4.1.2. Inferring step.

Source speech is converted into a sequence of PPPs by SI-ASR, then mapped into a sequence of MCEPs by DBLSTM. Additionally, the F0, AC and energy term of the MCEP features are extracted from the source speech. We used STRAIGHT [36] for F0 and AC extraction, and SPTK for MCEP extraction. The MCEPs from DBLSTM and energy term of the MCEPs from the source speech are concatenated and converted into a spectrum using SPTK. AC is directly extracted from the source speech. Finally, STRAIGHT synthesizes the target speech from spectrum, linearly converted $\log$(F0), and AC.

4.2. Intu

In this section, we describe operating process of the Intu modules that comprise the echoing model in detail. We used Intu installed on a PC running Windows 10 for our experiments.

4.2.1. Text extractor.

The text extractor converts wave data of type AudioData, sensed by an Intu device's microphone, into text data and registers the data in the blackboard. When Intu begins operation, the text extractor's OnStart method is called. The OnStart method receives sensor information for AudioData data from the sensor manager. During this process, the text extractor transmits the OnAddAudio method as a call-back function to the sensor manager. The OnAddAudio method takes a sensor object as an argument and subscribes to the AudioData data from the sensor. As a result, the text extractor is able to handle all data of type AudioData. In order to communicate with a sensor, the OnAddAudio method transmits the OnAudioData method as a call-back function. In the OnAudioData method, data of type AudioData is accepted as an argument and transmitted to Watson's STT service. The text data from Watson's STT service is wrapped in a Text type and registered in the blackboard.

4.2.2. Echo agent.

The echo agent is a module that converts data from type Text into type Say, which is subscribed by WinSpeech gesture. Although the model simply returns a user's speech in Intu's voice, the contents of the input and output speech should be treated differently in practice. First, the echo agent subscribes to Text data from the blackboard using an OnStart method. During this process, the echo agent sends an OnEcho method to the blackboard as a call-back function. When the OnEcho method is called, it extracts the Text data from the argument. The OnEcho method then wraps the data as Say type and registers it in the blackboard.

4.2.3. WinSpeech gesture.

As a default, a speech gesture exists in Intu to output speech. We overrode the class with WinSpeech gesture for Windows OS. When Say data is registered in the blackboard, the StartSpeech, OnSpeechData, and PlayStreamedSound methods are called in order to process the data. The StartSpeech method internally calls Watson's TTS service to convert text data into wave data. It then transmits the OnSpeechData method as a call-back function. The OnSpeechData method takes the converted wave data as an argument. It then pauses the data stream that the WinSpeech gesture manages and sensors for AudioData data. Within the OnSpeechData method, the PlayStreamedSound method is transmitted to the ThreadPool instance in order to play stored wave data on a thread. In the PlayStreamedSound method, streamed sound stored in the WinSpeech gesture is replaced with converted speech data through communication with the VCN server. This routine is terminated when the converted speech data are played over the speaker by

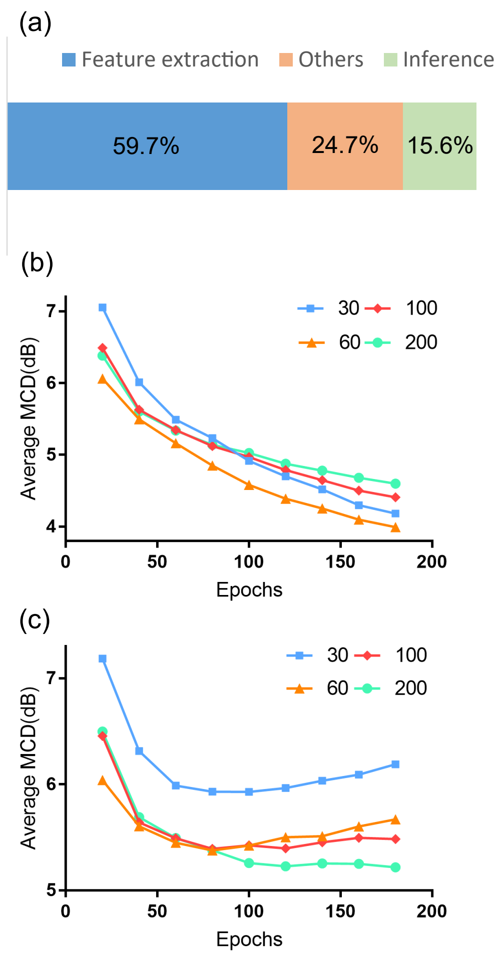

Figure 6. (a) Percentage of elapsed time, (b) graph of average MCD using training sets and (c) a validation set. (This figure should be viewed in color.)

the PlayStreamedSound method.

5. Results and discussion

We designed experiments to simulate real service situations. We measured the elapsed time for VCN processing and found the proper amount of target speech data for training DBLSTM.

In order to calculate the delay time from a user's point of view, the time consumed by Intu should also be considered. However, it is difficult to generalize the real circumstances because the developers may include various extra process routines when customizing Intu's structure. Therefore, we measured only the additional time consumed by VCN, excluding the time required to communicate with Intu.

In order to find the proper amount of data for training DBLSTM, mel-cepstral distortion (MCD) was used as a metric to evaluate voice conversion performance. We fed target speech into VCN and then measured the MCEP differences between the input and output speech. The formula for MCD is:

$$MCD(dB) = \frac{10}{\ln 10}\sqrt{2\sum_{d=1}^{D}\left(c_d - c_d^{converted}\right)^2} \quad (7)$$

where c_d is the d-th element of the MCEP label used for DBLSTM training and $c_d^{converted}$ is the d-th element of the output of DBLSTM. We excluded the energy term in this calculation.

5.1. Voice conversion time measurement

We divided the entire process into three steps: (1) PPP feature extraction by SI-ASR, (2) MCEP feature extraction by DBLSTM and (3) speech synthesis by the STRAIGHT vocoder. We then measured the time required for each step. The results are presented in Figure 6(a).

The total elapsed time for voice conversion is close to a minute. The most time-consuming part is the process of extracting the F0 feature using the STRAIGHT vocoder in step three, which accounts for 54.9% of the total time. The entire feature extraction process accounts for 59.7% of the total time, meaning that the time delay due to feature extraction should be focused on first. Additionally, the time required for loading the Matlab engine to run the STRAIGHT script in step three and performing the DBLSTM inference in step two account for 14.8% and 9% of the total time, respectively. The remaining processes account for 16.5% of the total time.

5.2. Size of the DBLSTM training set

The CMU ARCTIC [37] corpus was used as the training dataset for DBLSTM. A US female speaker called SLT was adopted from the corpus as the target voice. The version number of the corpus is 0.95. We set the number of speech samples for the target voice to 30, 60, 100, and 200 for DBLSTM training. The sizes of the mini-batches during training were 10, 10, 25, and 50, respectively. The learning rates were all set to 0.001. The training processes for SI-ASR are the same.

We measured the average MCD between MCEPs, one of which was extracted directly from the target speech, while the other was the output of DBLSTM using the same target speech as input. The graphs in Figures 6(b) and 6(c) plot the average MCD measurement against the size of the training set. DBLSTM was implemented in Tensorflow. An NVIDIA

Titan X Pascal was used for processing the neural network.

Figure 6(b) plots the average MCD measurements for different sizes of training sets against the number of epochs used for training. As the number of epochs increases, the average MCD is lower when the size of the training set is 30 or 60 than when it is 100 or 200. In contrast, the graph in Figure 6(c) plots the results of the same experiment when using a validation set for MCD estimation. The average MCD for the training set sizes of 30 and 60 decreases and then increases again as the number of epochs increases. Overall, the general performance degrades when the size of the training set is 30 or 60. However, when DBLSTM is trained with 100 or more data samples, the average MCD for the validation set tends to decrease as the number of epochs increases.

5.3. Discussion

Through the experiment described in section 5.1, we analyzed how to minimize the time delay in order to facilitate commercialization of our voice customization service. The first method is to remove unnecessary time delays. For instance, the percentage of time required for loading the Matlab engine was 14.8%. This can be removed because the Matlab code of the STRAIGHT vocoder can also be implemented in python, preventing the unnecessary loading of a separate engine.

Additionally, steps one and two can be performed independently of the process for extracting features from the source speech in step three. If the two processes are performed in parallel, the time required to complete both is reduced to that of the longer of the two. In our case, the two processes account for 26.1% and 72.6% of the total time, meaning the total time can be reduced from 98.7% to 72.6% of the total time.

The final method is based on the fact that IBM's TTS model stores the acoustic units of the source speech in a database [7]. PPP, F0, MCEP, and AC can all be extracted and stored in the database for the model in advance. Watson's TTS server then returns frame-wise acoustic features corresponding to the input text. This means that, in the VCN, there is no need for step one or the feature extractions in other steps during the inference stage.

If all of these methods are successfully applied, the expected total time would be reduced to approximately 10 seconds. We expect that any other optimizations in computation and pipelining will further reduce processing time.

Based on our experiments, we conclude that it is appropriate to ask a user for more than 100 speech data samples for the target voice. When we consider the dataset we used in the experiments, the total duration of speech that the user must prepare would be less than 10 min for 100 utterances.

6. Conclusion

In this study, we proposed a voice customization service as a method of enhancing the user experience with IBM Project Intu, which is an intelligent personal assistant service based on IoT. Our model combines a conventional TTS model with the voice conversion model proposed in [13] in order to reduce the burden of preparing training data. From our experimental results, we determined which parts of the process consume the most time and the quantity of target speech samples required to perform the voice customization. We also discussed areas for improvement and methods to optimize the performance of the voice customization service. We expect that our research will provide an excellent basis for voice customization in intelligent personal assistant services.

7. Acknowledgement

This work was supported by the University Program of IBM Korea. J. Song, J. Lee, H. Kim, and S. Yoon were also supported in part by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2016-0-00087, Development of HPC System for Accelerating Large-scale Deep Learning), in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (No. 2016M3A7B4911115), and in part by the Brain Korea 21 Plus project in 2017. This material is based upon work supported by an IBM Faculty Award.

References

[1] "Amazon Alexa." https://developer.amazon.com/alexa.

[2] "Google Home." https://madeby.google.com/intl/en_us/home/.

[3] "IBM Project Intu." https://www.ibm.com/watson/developercloud/project-intu.html.

[4] "IBM Watson." https://www.ibm.com/watson/.

[5] "IBM Watson: Conversation." https://www.ibm.com/watson/developercloud/doc/conversation/index.html.

[6] "IBM Watson: Language Translator." https://www.ibm.com/watson/developercloud/doc/language-translator/index.html.

[7] "IBM Watson: Text-to-Speech." https://www.ibm.com/watson/developercloud/doc/text-to-speech/index.html.

[8] A. van den Oord, S. Dieleman, H. Zen, *et al.*, "Wavenet: A generative model for raw audio.," *arXiv preprint arXiv:1609.03499*, 2016.

[9] Z. Wu, E. Chng, H. Li, *et al.*, "Conditional restricted Boltzmann machine for voice conversion.," in *Signal and Information Processing (ChinaSIP), 2013 IEEE China Summit & International Conference on*, pp. 104–108, IEEE, 2013.

[10] T. Nakashika, R. Takashima, T. Takiguchi, *et al.*, "Voice conversion in high-order eigen space using deep belief nets.," pp. 369–372, Interspeech, 2013.

[11] L. Sun, S. Kang, K. Li, *et al.*, "Voice conversion using deep bidirectional long short-term memory based recurrent neural networks.," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pp. 4869–4873, IEEE, 2015.

[12] T. Nakashika, T. Takiguchi, and Y. Ariki, "Voice conversion using RNN pre-trained by recurrent temporal restricted Boltzmann machines.," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 23, no. 3, pp. 580–587, 2015.

[13] L. Sun, K. Li, H. Wang, *et al.*, "Phonetic posteriorgrams for many-to-one voice conversion without parallel data training.," in *Multimedia and Expo (ICME), 2016 IEEE International Conference on*, pp. 1–6, IEEE, 2016.

[14] H. Zen, K. Tokuda, and A. Black, "Statistical parametric speech synthesis.," *Speech Communication*, vol. 51, no. 11, pp. 1039–1064, 2009.

[15] "Wikipedia: Speech synthesis." https://en.wikipedia.org/wiki/Speech_synthesis.

[16] "IBM Watson Text-to-Speech: The science behind the service." https://www.ibm.com/watson/developercloud/doc/text-to-speech/science.html.

[17] T. Yoshimura, K. Tokuda, T. Masuko, *et al.*, "Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis.," in *Sixth European Conference on Speech Communication and Technology*, pp. 2347–2350, 1999.

[18] S. Arik, M. Chrzanowski, A. Coates, *et al.*, "Deep voice: Real-time neural text-to-speech.," *arXiv preprint arXiv:1702.07825*, 2017.

[19] R. Fernandez, A. Rendel, B. Ramabhadran, *et al.*, "Using deep bidirectional recurrent neural networks for prosodic-target prediction in a unit-selection text-to-speech system.," in *Sixteenth Annual Conference of the International Speech Communication Association*, pp. 1606–1610, Interspeech, 2015.

[20] R. Fernandez, A. Rendel, B. Ramabhadran, *et al.*, "Prosody contour prediction with long short-term memory, bi-directional, deep recurrent neural networks.," pp. 2268–2272, Interspeech, 2014.

[21] M. Abe, S. Nakamura, K. Shikano, *et al.*, "Voice conversion through vector quantization.," *Journal of the Acoustical Society of Japan (E)*, vol. 11, no. 2, pp. 71–76, 1990.

[22] M. Savic and I. Nam, "Voice personality transformation.," *Digital Signal Processing*, vol. 1, no. 2, pp. 107–110, 1991.

[23] Y. Stylianou, O. Cappé, and E. Moulines, "Continuous probabilistic transform for voice conversion.," *IEEE Transactions on speech and audio processing*, vol. 6, no. 2, pp. 131–142, 1998.

[24] A. Kain and M. Macon, "Spectral voice conversion for text-to-speech synthesis.," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, vol. 1, pp. 285–288, IEEE, 1998.

[25] T. Toda, H. Saruwatari, and K. Shikano, "Voice conversion algorithm based on Gaussian mixture model with dynamic frequency warping of STRAIGHT spectrum.," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, vol. 2, pp. 841–844, IEEE, 2001.

[26] Y. Chen, M. Chu, E. Chang, *et al.*, "Voice conversion with smoothed GMM and MAP adaptation.," in *Eighth European Conference on Speech Communication and Technology*, pp. 2413–2416, 2003.

[27] H. Hwang, Y. Tsao, H. Wang, *et al.*, "Alleviating the over-smoothing problem in GMM-based voice conversion with discriminative training.," pp. 3062–3066, Interspeech, 2013.

[28] T. Toda, A. Black, and K. Tokuda, "Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory.," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 8, pp. 2222–2235, 2007.

[29] Z. Wu, T. Virtanen, E. Chng, *et al.*, "Exemplar-based sparse representation with residual compensation for voice conversion.," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 10, pp. 1506–1521, 2014.

[30] L. Chen, Z. Ling, Y. Song, *et al.*, "Joint spectral distribution modeling using restricted Boltzmann machines for voice conversion.," *Interspeech*, pp. 3052–3056, 2013.

[31] "Reference manual for speech signal processing toolkit ver. 3.10." http://sp-tk.sourceforge.net/.

[32] D. Povey and G. Saon, "Feature and model space speaker adaptation with full covariance Gaussians.," pp. 1145–1148, Interspeech, 2006.

[33] "IBM Watson: Speech-to-Text." https://www.ibm.com/watson/developercloud/doc/speech-to-text/index.html.

[34] "The DARPA TIMIT acoustic-phonetic continuous speech corpus (TIMIT)." https://catalog.ldc.upenn.edu/docs/LDC93S1/timit.readme.html.

[35] D. Povey, A. Ghoshal, G. Boulianne, *et al.*, "The Kaldi speech recognition toolkit.," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFL-CONF-192584, IEEE Signal Processing Society, 2011.

[36] H. Kawahara, I. Masuda-Katsuse, and A. De Cheveigne, "Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based f0 extraction: Possible role of a repetitive structure in sounds.," *Speech communication*, vol. 27, no. 3, pp. 187–207, 1999.

[37] J. Kominek and A. Black, "The CMU arctic speech databases.," in *Fifth ISCA Workshop on Speech Synthesis*, pp. 223–224, 2004.