2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)

Automated Testing of Basic Recognition Capability for Speech Recognition Systems

Futoshi Iwama
IBM Research
IBM Japan
Tokyo, Japan
gamma@jp.ibm.com

Takashi Fukuda
IBM Research
IBM Japan
Tokyo, Japan
fukuda1@jp.ibm.com

***Abstract*—Automatic speech recognition systems transform speech audio data into text data, i.e., word sequences, as the recognition results. These word sequences are generally defined by the language model of the speech recognition system. Therefore, the capability of the speech recognition system to translate audio data obtained by typically pronouncing word sequences that are accepted by the language model into word sequences that are equivalent to the original ones can be regarded as a basic capability of the speech recognition systems. This work describes a testing method that checks whether speech recognition systems have this basic recognition capability. The method can verify the basic capability by performing the testing separately from recognition robustness testing. It can also be fully automated. We constructed a test automation system and evaluated though several experiments whether it could detect defects in speech recognition systems. The results demonstrate that the test automation system can effectively detect basic defects at an early phase of speech recognition development or refinement.**

***Keywords*-Speech Recognition System, Automated Testing, Modeling of ASR System Testing**

I. Introduction

Automatic speech recognition (ASR) systems are increasingly being used in practical applications, and there is a corresponding demand for new systems, including both general-purpose ASR and domain-specific ASR. For example, ASRs in smart phones are popular general-purpose ASRs designed for handling speech data with a wide range of content in daily use. Domain-specific ASRs are also starting to be used simultaneously or integrated into general-purpose ASRs for recognizing audio data obtained from speech uttered in specific situations or tasks, such as at ticket vending machines, during voice dialing, or in the ASR module of smart speakers. Consequently, it would be an important software engineering project to study ASR system testing, which is still dependent on individual skill in many cases, and to create more automatic/systematic test methods for ASR systems.

In general, recognition capability testing of ASR systems is used to confirm that each targeted audio data is definitely transformed into text data, just as designed. These tests have to handle various kinds of audio data depending on the characteristics of the speech sound (e.g., female/male or adult/child) and environment (e.g., in a car or on the phone) where ASR is used. In brief, we must check that this variety of audio data can be robustly recognized with satisfactory accuracy by the ASR system being tested. Such tests are usually regarded as recognition robustness tests and are a critical (and maybe the only well-defined) part of practical ASR testing [1].

At this time, the robustness test is usually difficult to automate, which makes it a labor-intensive task. This is partly because it requires collecting various kinds of audio data for estimating the robustness. Moreover, the robustness test can only be performed for low coverage of target words and sentences that are defined in and can be handled by ASR, since the number of target words and sentences is usually enormous and it is impossible to collect various speech data for each word and sentence. For example, a typical general-purpose ASR sytem defines more than 100,000 words in a lexicon of ASR. Therefore, in the robustness test, many target words and sentences are left unconfirmed. For example, if the lexicon of an ASR system does not contain required words (such as the names of a station, town, or person), the defects may be left untested and the ASR may be released without the capability of recognizing these names. Another example of a defect that is prone to be left unconfirmed without being detected in the robustness test is where the pronunciation "dɪˈrekʃ(ə)n" is registered for the word "`direction`" but "daɪˈrekʃ(ə)n" is not for the same word in the lexicon. When ASR systems contain such defects, they cannot recognize sentences involving those words and pronunciations under any ideal circumstances. Generally, this type of defect can be thought of as separate from ones related to recognition robustness.

To examine these types of defects with high coverage — in other words, to test that ASR has the minimum capability to correctly recognize a large amount of target words and sentences — it is necessary to collect a huge amount of *typical* speech data. In the current development of speech recognition systems, this kind of test is usually being conducted unsystematically as just one part of the robustness test, without being distinguished from the robustness test. Moreover, unfortunately, much of the research in the speech recognition field focuses on improving the recognition robustness, and it seems that systematic/automated ASR

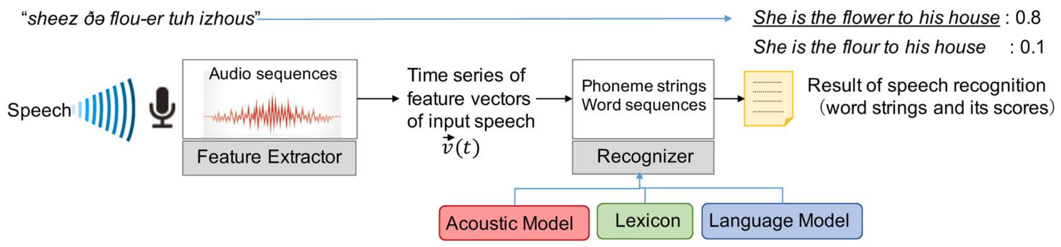

"sheez ðə flou-er tuh izhous"
She is the flower to his house : 0.8
She is the flour to his house : 0.1
Speech
Audio sequences
Feature Extractor
Time series of feature vectors of input speech $\vec{v}(t)$
Phoneme strings
Word sequences
Recognizer
Result of speech recognition (word strings and its scores)
Acoustic Model
Lexicon
Language Model

Figure 1. Overview of ASR system: Components and model data of ASR system (acoustic model, lexicon, and language model).

testing method has not been extensively studied. Perhaps due to these situations, the above type of defects occur very frequently when developing and extending an ASR system.

In this paper, we present a test viewpoint for confirming the basic recognition capability of an ASR system and construct an automated test method, which we call a *basic recognition test*, on the basis of this viewpoint. Intuitively, the test viewpoint is to verify whether the ASR system under test (ASR-SUT) has the capability to recognize typical speech for target words and sentences under ideal conditions by separating it from recognition robustness testing. The proposed test method can automatically detect words and sentences that are never recognized due to functional defects of the ASR. Especially, because the test method is fully automated, it can verify whether *every* word defined in ASR-SUT can be recognized at least in an ideal situation.

The two main contriutions of this paper are that it (C1) formalizes the basic recognition test for ASR and constructs an automated test system for the basic recognition test, (C2) shows that functional defects of lexicon and acourstic model of ASR can be automatically detected by using the automated test system. We adopt the following approaches to materialize these contributions:

(I) Comparing formally a conventional test (including verification of recognition robustness) and the basic recognition test for ASR systems.
(II) Constructing the automated test system by (a) generating test sentences from the language model of the ASR-SUT with coverage criteria, (b) synthesizing a set of audio data files from each test sentence by using multiple text-to-speech (TTS) synthesizers with different speech characteristics, and (c) enabling the ASR-SUT to recognize every audio data file in the set to obtain a set of recognition results and verifying that the resulting word sequences are *as a whole* equal to the original test sentence.

Note that, as with the basic recognition test for ASRs, functional tests on machine-learning-based software suffer from the same problem as above, and are often difficult to be systematically performed or automated with high coverage. This is because, for such type of software, no correct output may be obtained for each input in advance [2], [3], and because high coverage test input set is often difficult to be systematically obtained. Our method can be thought of as a special case of automation of a functional test on machine-learning-based software, which is specialized for ASR. In the ASR cases, we found that the test automation can be realized because test input speech which may be regarded as typical under a specific equality judgement can be generated from each possible output with multiple TTS synthesizers.

Section II of this paper introduces the basic recognition test for ASR. Section III describes a test automation system for the basic recognition test. Section IV explains how we evaluated the test automation system through several experiments and Section V discusses some practical aspects of the test automation. Section VI describes related work and we conclude in Section VII with a brief summary and mention of future work.

II. Basic Recognition Test for ASR

This section first outlines the processing and components of ASR systems from the software engineering perspective and then introduces and formalizes the test viewpoint for checking the basic recognition capabilities of ASR.

A. Overview of ASR system

An overview of the components and model data used in the ordinal ASR systems and their processing is shown in Fig.1. On the whole, speech is first digitalized as time sequential signal data (called *audio data* or *audio sequences* here) and transformed by the feature extractor component into a time series of feature vectors. After that, the feature vectors are transformed into word sequences on the recognition language of ASR. Each resulting word sequence has a score that indicates the probability of the results, and the word sequence with the best score becomes the speech recognition output. For example, suppose we make an ASR system process speech like " sheez ðə flou-er tuh izhous" . It transforms this speech into several candidate sentences with scores such as `[She is the flower to his house: 0.8]` and `[She is the flour to his house: 0.1]`. The system outputs the first text as the result because it has the best score.

The recognizer component in Fig.1 is also called a *decoder*. It transforms the time series of feature vectors into the resulting word sequence, and its process consists of three models: an *acoustic model*, which involves stochastic mapping from phonemes to the time series of feature vectors, a *lexicon*, which contains information on stochastic mapping from words to phoneme strings (indicating the pronunciations of each word), and a *language model*, which expresses the grammar of the recognition text (i.e., word sequences) and the probability of each text matching the grammar. Here, we use meta-variables v, x, and w to range over the time series of feature vectors, phoneme sequences, and word sequences, respectively. We can then express the acoustic model, lexicon, and language model as the (conditional) probabilities $P_A(v|x)$, $P_D(x|w)$, and $P_L(w)$. The decoder actually computes the word sequence $\hat{w}$, given by the following equation, which maximizes the posteriori probability $P_A(v|x)P_D(x|w)P_L(w)$ by simultaneously considering the three models at the same time [4]:

$$\hat{w} = \underset{w}{argmax}\ P_A(v|x)P_D(x|w)P_L(w). \tag{1}$$

Acoustic models are usually implemented by hybrid usage of hidden markov models (HMMs) [5] and neural networks [6]. Lexicons can be implemented by using a possibility that each word w is pronounced by x, such that the probability of pronouncing ”ɪnkwəɪri” for the word ”Inquiry” is 0.8, and that of pronouncing ”ɪnkwəri” is 0.2. Language models for domain-specific ASRs are usually implemented by probability-weighted finite state automatons (WFST) [7]. We usually use a stochastic model for general-purpose ASRs, such as an N-gram [8] (that actually can be expressed by WFST), to implement language models because the ASRs must handle a wide range of spoken sentences.

These three models need to be created, revised, and extended as an ASR system is developed, modified, and expanded. Therefore, we generally prepare these as replaceable models. For example, acoustic models are to be revised depending on the characteristics of speech sound handled by ASR, such as female/male/adult/child sounds. Lexicons are extended as ASR needs to handle new words or pronunciations, and language models are also revised when we need to change the probability of a recognition sentence with the grammar, e.g., as new words are added to the grammar. In these extensions, if necessary data is not added, or if defective data is added, the ASR system will experience defects in the basic recognition capability.

B. Basic Recognition Test

We first formalize the normal testing currently performed in ASR developments. Next, we formalize the basic recognition testing and then compare the two tests.

We use the meta-variable S to indicate an ASR system under test. Possible recognition result texts for S are given by a set of word sequences that are accepted by the language model P_L of S with probabilities greater than a small constant ϵ. We call the set of word sequences a *recognition language* (each element of the set is called a recognition text or sentence or word sequence) and write $\mathcal{L}(S,\epsilon)$ for the language. Given a probability constant ϵ, the recognition language $\mathcal{L}(S,\epsilon)$ of S is defined by

$$\mathcal{L}(S,\epsilon) = \{w \mid P_L(w) > \epsilon,\ P_L \text{ is the language model of } S\}.$$

1) Formalization of Positive Test Data: We use the terms *positive (speech) audio sequences* and *positive word sequences* for S to indicate audio data that are to be correctly recognized by S for the former and word sequences obtained by correctly writing out positive speech audio data for S for the latter to consider descriptive specifications of speech audio data to be correctly recognized. We also write $\Omega(S)$ for the set of all possible *positive* word sequences for S. The set $\Omega(S)$ can be regarded as an ideal descriptive specifications of which speech data should be correctly recognized by S. Therefore, if the language model is consistently defined according to the specification of S, all recognition word sequences for S are positive word strings for S because audio data obtained by correctly converting recognition texts into speech should be correctly recognized by S. Therefore, for such consistent language models, we have

$$\mathcal{L}(S,\epsilon) \subseteq \Omega(S). \tag{2}$$

The equality relation in the above relation may not hold, not only due to the incompleteness of the language model with regard to the ideal specification, but also spelling variants.

Here, let us consider ASR system S which is integrated into a car-navigation system and therefore is only used in cars. We then *ideally* need to collect all positive audio data that are obtained in cars. However, even if one person speaks the same text w twice in the same manner under the same restricted condition, we actually cannot obtain two speech audio data that have the same sound wave shapes. Therefore, the size of a set containing all positive audio data can be limitless in real situations.

In order to model the set of all positive audio data for S, we consider and assume the set of all positive audio data obtained by correctly uttering the positive word sequence w. We also assumed a conceptualistic function $\mathbf{Sp}[S](\cdot)$ that maps each positive word sequence w for S to the set of all positive audio data of w for S. The domain of $\mathbf{Sp}[S]$ has to include $\Omega(S)$. We call the function $\mathbf{Sp}[S]$ *speech oracle* for S. The set of all positive audio data for S is given by

$$\bigcup_{w\in\Omega(S)} \mathbf{Sp}[S](w). \tag{3}$$

This is because there is at least one positive audio sequence for every positive word sequence w. Hence, we have

$$\forall w \in \Omega(S).\ \mathbf{Sp}[S](w) \neq \phi, \tag{4}$$

where ϕ is the empty set.

We use meta-variables d and $\mathcal{D}$ for expressing an audio sequence and a set of audio sequences, respectively. As every positive audio sequence is contained in the set given by (3), every positive audio sequence can be expressed as an index notation by using a positive word sequence. We use the index notations d_w and $\mathcal{D}_w$ for d and $\mathcal{D}$ that satisfy $\mathbf{Sp}[S](w) \supseteq \mathcal{D} \ni d$ and $w \in \Omega(S)$. We call d_w and $\mathcal{D}_w$ an audio sequence (set) *according to word sequence* w. We can regard d_w and $\mathcal{D}_w$ as positive test input data (set) with expectation value w. We can find an expectation value for every positive input.

We first collect sample positive speech data for S in many actual cases of ASR tests and then create a test set of positive word sequences $W \subseteq \Omega(S)$ based on the samples (and other background usage information of S). Next we prepare an actual test input set of various audio data for each positive word string in the set W. Because there is actually no such function as the speech oracle $\mathbf{Sp}[S]$ in the last step that automatically provides a variety of positive audio data according to each given word sequence, we must create and collect an adequate number of positive audio data, $\mathcal{D}_w \subseteq \mathbf{Sp}[S](w)$, for each $w \in W$.

2) Formalization of ASR Tests: Here, we formalize both a test for checking recognition robustness (called a *positive-data recognition test here*) and the new test for ASR, namely, the *basic recognition test*.

Definition 1 (ASR System S): The ASR system under test S consists of a function $\mathbf{ext}(\cdot)$ that transforms each speech audio sequence into a time series of feature vectors, an acoustic model P_A, a lexicon P_D, a language model P_L, and a function $\mathbf{dec}(\cdot)$ that maps each feature vector series into a recognition word sequence according to Eq. (1).

Functions $\mathbf{ext}(\cdot)$ and $\mathbf{dec}(\cdot)$ above correspond to the feature extractor and recognizer in Fig.1. As most of state-of-the-art ASR systems have this configuration, we formalized the two tests for ASR systems following the above definition.

We also assumed *equality judgments on word sequences* $dpm(\cdot,\cdot)$ to check whether two word sequences could be considered to be the same. Such judgments are actually implemented by using dynamic programming (DP) matching [9]–[11] through the use of various string and phoneme distance measures depending on the intended end-usage for ASR systems. For example, a dpm_1 behaves like dpm_1(“BSD license”, “BSD licence”)=*true*, but another dpm_2 may behave like dpm_2(“Please cheque”, “Please check”)=*false*. In some languages, including Japanese, there may be several descriptions for one word. For example, the pronunciation “sakura” (meaning cherry blossom) may be written as “サクラ” or “桜”. Therefore, it needs to be understood that we need to prepare equality judgments such that dpm_3(“サクラ”,“桜”) = *true* for testing an ASR system for Japanese. From here, we fix the word-sequence equality judgment dpm and write $w_1 \cong w_2$ if and only if $dpm(w_1, w_2) = true$.

Here, the positive-data recognition test for S — a commonly used test for ASR systems — is formalized as follows:

Definition 2 (Positive-Data Recognition Test): Given S, select enough $W \subseteq \Omega(S)$ and, for each $w \in W$, collect enough non-empty $\mathcal{D}_w \subseteq \mathbf{Sp}[S](w)$. Then verify that the following holds:

$$\forall w \in W \subseteq \Omega(S).\ \forall d_w \in \mathcal{D}_w \subseteq \mathbf{Sp}[S](w).\ (\mathbf{dec}(\mathbf{ext}(d_w)) \cong w).$$

In the test, we first select W, i.e., an adequate number of positive word sequences for S, and then prepare sufficient test input sets $\mathcal{D}_w$ for each $w \in W$ by collecting various positive audio data according to w. We then verify that every positive audio sequence $d_w \in \mathcal{D}_w$ can be recognized as the same word string as w.

Note that this test includes checking for recognition robustness and accuracy because it verifies that various positive audio data can be correctly recognized for each test sentence. It can actually produce data for evaluating recognition robustness/accuracy. For example, word error rate *WER*, which is a commonly used metric for the performance of speech recognition, can be calculated for each test sentence w by

$$WER(w) = (S(w) + D(w) + I(w)) \ / \ N(w), \tag{5}$$

where $S(w)$ / $D(w)$ / $I(w)$ are the numbers of wrong substitution / deletion / insertion words of the recognition results $\mathbf{dec}(\mathbf{ext}(d_w))$ compared with correct sentence w up to the relation $\cong$. $N(w)$ is the number of words in the correct sentence. For example, by using w = “BSD licence is applied to this software”, $\mathbf{dec}(\mathbf{ext}(d_w))$ = “BSE license is applied to software”, and “license” $\cong$ “licence”, we obtain $WER(w) = ((1 + 1 + 0)/7 = 0.2857\cdots$, where “this” is the deletion word. Recognition robustness is usually evaluated by classifing audio sequences $\mathcal{D}_w$ into groups according to attributes (e.g., kind of task, gender, age, and type of sentence w) and then calculating the average *WER* for each group. The evaluations and improvements in recognition robustness are seem to remain as unsystematic task based on expert knowlege.

The basic recognition test for the ASR system is now defined as the following:

Definition 3 (Basic Recognition Test): Given S, generate enough $W \subseteq \mathcal{L}(S,\epsilon)$. Then verify that the following holds:

$$\forall w \in W \subseteq \mathcal{L}(S,\epsilon).\ \exists d_w \in \mathbf{Sp}[S](w).\ (\mathbf{dec}(\mathbf{ext}(d_w)) \cong w).$$

This test verifies that at least one positive audio d_w sequence is converted into a sentence that is equivalent to w by S. In other words, the viewpoint behind this test only checks that S has the capability to correctly recognize at least one audio sequence, which may be the most typical one, for each generated recognition sentence w. Therefore,

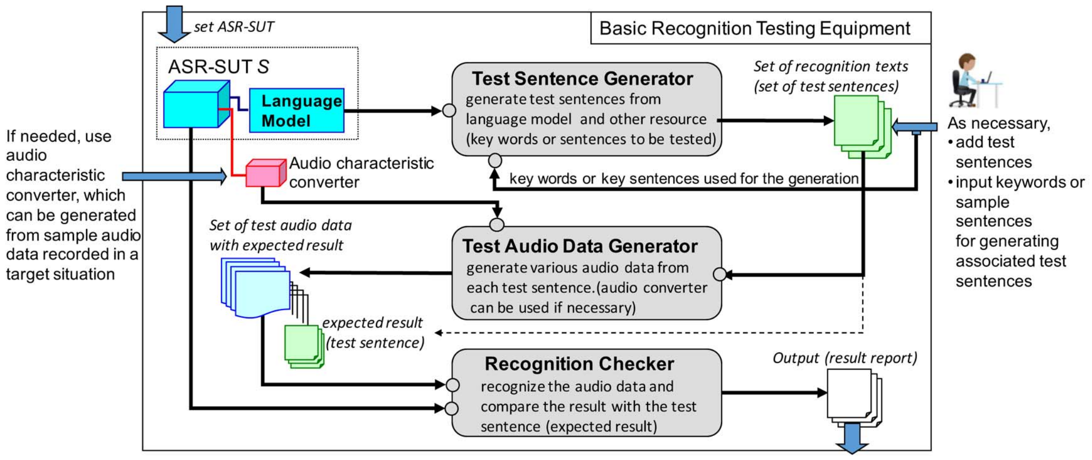

set ASR-SUT
Basic Recognition Testing Equipment
ASR-SUT S
Language Model
Test Sentence Generator
generate test sentences from language model and other resource (key words or sentences to be tested)
Set of recognition texts (set of test sentences)
As necessary,
• add test sentences
• input keywords or sample sentences for generating associated test sentences
If needed, use audio characteristic converter, which can be generated from sample audio data recorded in a target situation
Audio characteristic converter
key words or key sentences used for the generation
Set of test audio data with expected result
Test Audio Data Generator
generate various audio data from each test sentence.(audio converter can be used if necessary)
expected result (test sentence)
Recognition Checker
recognize the audio data and compare the result with the test sentence (expected result)
Output (result report)

Figure 2. Configuration of basic recognition test.

it is separated from the viewpoints of recognition robustness, which are related to checking that various kinds of audio data according to the same test sentence are correctly recognized.

Under the premise that S has a correct language model satisfying (2), the following holds:

Lemma 4 (Necessity for basic recognition test): For the same selected $W \subseteq \mathcal{L}(S, \epsilon)$, passing the basic recognition test is a necessary condition for passing the positive data recognition test under the same word sequence equality $\cong$.

This is formally shown using the equations (2), (4) and the definitions 2, 3. Hence, the basic recognition test can be a necessary condition of the positive data recognition test, which includes the robustness test. However, in the basic recognition test, we do not need to gather speech data for each test text w if we can generate typical speech data for w. Therefore, in this case, the basic recognition test can be automated and thereby handle an enormous amount of test sentences. The automated method of the basic recognition test is described in the next section.

C. Detectable and Undetectable Defects

Let us assume that we have a correct language model. When S fails the basic recognition test for a test sentence, the defects are probably in some parts of software components related to the test sentence. The following defective parts and causes in this case are considered to be:

F1. Lexicon contains missing words or erroneous data about how to pronounce words in the test sentences,

F2. Acoustic Model contains missing or erroneous data for phonemes related to words in the sentences, and

F3. Decoder Program contains bugs or incorrect parameters in parts for processing word sequences in the sentences.

On the other hand, the following cannot be checked from the test viewpoint of the basic recognition test:

N1. Whether ASR has a correct language model that accepts positive texts and, rejects negative ones.

N2. Whether ASR do not recognize speech that is not permitted to be recognized (negative data test).

N3. Whether ASR has sufficient recognition robustness for various types of utterances.

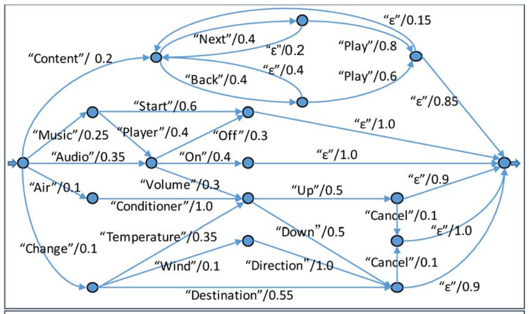
The basic recognition test assumes the correctness (i.e., (2)) of the language model (N1) and is assumed for testing negative data (N2) and recognition robustness (N3). Therefore, the basic recognition test is suitable as a preliminary test step for the existing recognition robustness test on ASR systems. Note that the existing robustness testing of ASR systems also assumes the correctness of the language model.

In this section, we proposed a test viewpoint for ASR that checks defects related to the basic recognition capability given by Definition 3 separately from testing robustness of recognition, which are conventionally checked together with the basic capability in a similar form as the positive data recognition test given by Definition 2.

III. Automation of Basic Recognition Test

This section describes a system for automating the basic recognition test. An overview of the configuration for the system is given in Fig.2. Roughly speaking, it first generates test sentences from the language model of ASR-SUT or from some other positive sentence resource, e.g., collections of speech processed by previous system versions of ASR-SUT. It then generates a set of audio data from each of the generated test sentences. Here, we use multiple TTSs with different speech characteristics for generating various positive audio data. It finally recognizes these audio data by using ASR-SUT itself to generate a set of recognition texts for each test sentence and checks that the resulting set is, as a whole, equal to the test sentence. There are three main modules in the configuration, discussed in more detail below.

The test sentence generator module generates test sentences that are accepted by the language model of the ASR-



The following word sequences **s1…s11** are generated as test sentences in all edges criterion from the above language model given by WFST:

s1. ["Change" "Temperature" "Up" ""], **s2.** ["Change" "Destination" "Cancel" ""]
s3. ["Change" "Wind" "Direction" "Cancel" ""], **s4.** ["Air" "Conditioner" "Down" ""]
s5. ["Air" "Conditioner" "Up" ""], **s6.** ["Audio" "Volume" "Up" "Cancel" ""]
s7. ["Music" "Start"], **s8.** ["Music" "Player" "Off" ""],
s9. ["Music" "Player" "On" ""], **s10.** ["Content" "Next" "Play" "" "Next" "Play" ""]
s11. ["Content" "Back" "" "Next" "" "Back" "Play" ""]

Figure 3. Example of language model and test sentences.

SUT. When the language model is given by a relatively small size grammar expressing predefined or restricted domain-specific sentences, we can generate effective test sentences with coverage criteria from only the language model by using conventional techniques to analyze finite state automaton [12]. For example, the ASRs used in ticket vending machines or speech control systems for home electronics are in this category. We have presented an example, which is (part of) a language model of the ASR given by WFST and the set of test sentences generated from the model, in Fig. 3. Each label "xxx"/n on the edges in the figure means that the probability of transition obtained by word "xxx" is n. Word sequences s1...s11 are test sentences that cover all edges of the finite state transducer.

When the language model is given by a large stochastic model such as an N-gram that contains too many words, it may be useful for the test sentence generator to receive keywords or sub-sentences that are required to be specifically tested. For example, important proper names, technical terms, frequently appearing phrases, and compliance related words (e.g., "absolutely" and "agreement") often require intensive checking to be selected as the input keywords/sentences. The generator generates test sentences containing the input keywords/sub-sentences. We can use several techniques or heuristics for generating sentences from large (stochastic) language models [13], [14]. Whatever method we use in this module, we only need to prepare a program that generates effective test sentences that are accepted by the language model.

Because test statements generated by this module are then automatically processed at later steps, we can safely generate a large number of them. This drives the basic recognition test for high coverage target words and sentences.

The test audio data generator module generates a set of test audio data from each test sentence. A detailed configuration for this module is shown in Fig.4. Strictly

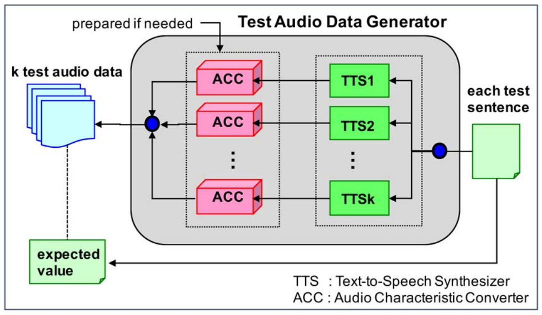

prepared if needed
Test Audio Data Generator
k test audio data
ACC
ACC
ACC
TTS1
TTS2
TTSk
each test sentence
expected value
TTS : Text-to-Speech Synthesizer
ACC : Audio Characteristic Converter

Figure 4. Configuration for test audio data generator.

speaking, the basic recognition test (of Definition 3) requires the speech oracle $\mathbf{Sp}[S]$ for S; however, no such oracle actually exists. Therefore, we take the following approach: *We represent the set of all positive audio data of w for S (i.e., $\mathbf{Sp}[S](w)$) by using typical audio data according to test sentence w and generate typical audio data that should be correctly recognized by S by using several TTS synthesizers with different sound characteristics.* TTS1⋯TTSk in Fig.4 are the prepared speech synthesizers to generate test audio data with different sound characteristics. We included k test audio data in this configuration for each test sentence generated by the test sentence generator module. The set of test audio data is to be translated into a set of recognized texts that are then compared as a whole with corresponding test sentences in the recognition checker module.

There are some cases where such typical audio data cannot be synthesized by any combination of common speech synthesizers due to specific environmental noise. For example, the ASRs used in car navigation systems need to handle speech under the acoustic condition inside cars. We can use an audio characteristic converter that converts audio data to those in special acoustic environments to address this problem in the manner shown in Fig. 4. Such a converter can actually be constructed from a small amount of speech data in the assumed acoustic environments [15], [16].

The recognition checker module first recognizes the test audio data by using ASR-SUT, then generates a set of recognized texts for each test sentence, and finally compares the set of recognition result texts with the test sentence to check whether they can be considered equal. Considering Definition 3 on whether there are audio data corresponding to each test sentence that are correctly recognized by ASR-SUT, the equality judgment is: *The set of recognized texts is considered equal to the test sentence if and only if, for each word subsequence of the test sentence, there exists an equal subsequence (w.r.t. $\cong$) of a recognized text in the set of the recognition texts*, rather than if there exists the same recognized text as the test sentence. For example, we can judge that the set containing two recognized word sequences "$\underline{\texttt{two tickets}}$ `to Saint Barbara`", "`to`

Algorithm 1: collectiveEqCheck (text,TxtSet)

```
Input: text:string and TxtSet:set of string
Output: true or false with unmatched parts
$WG \longleftarrow$ makeWordGraph(TxtSet) ;
if $WG$.containsPath(text) then return true;
else return false with matching part information;
```

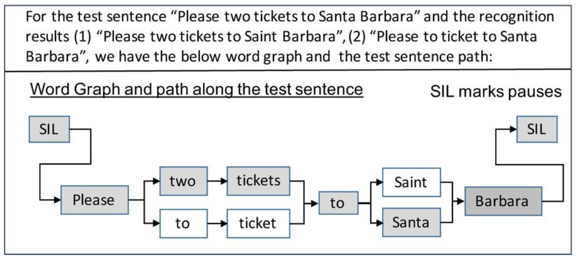

For the test sentence "Please two tickets to Santa Barbara" and the recognition results (1) "Please two tickets to Saint Barbara", (2) "Please to ticket to Santa Barbara", we have the below word graph and the test sentence path:

Figure 5. Example of word graph and path for test sentence.

`ticket` <u>`to Santa Barbara`</u>" is equal to the test word sequence "`two tickets to Santa Barbara`" since the underlined subsequences can comprise the test sentence and this implies that two parts of the audio data recognized as the underlined parts comprise the audio data that are correctly recognized by ASR-SUT.

We call this equality judgment for a set of recognition texts and a test sentence *collectiveEqCheck*. The equality is given by Algorithm 1, where `makeWordGraph`(X) makes a word graph, which considers pauses, from a set of recognition word sequences X, and WG.`containsPath`(t) checks whether there is a path along word sequence t in the word graph WG. An example of the word graph and the path (gray parts) along the given test sentence is shown in Fig. 5. Because there is a path along the test sentence in the graph, `collectiveEqCheck` returns true. The word graph is a common data structure for evaluating speech recognition results [10], [17], but the equality judgment by checking for the existence of a path for the test sentence seems to be specific to our test method.

The overall algorithm for the automated basic recognition test is given in Algorithm 2, where `Recognize`(S, d) means speech recognition for audio data d by using S that returns a text as the recognition result.

We assumed that multiple speech synthesizers in the automated test system could generate a set of test audio data that could be considered equal to typical audio data, which should be correctly recognized by ASR-SUT S (we will evaluate this assumption in the next section). The test system under this assumption uses multiple TTSs instead of the speech oracle, **Sp**$[S]$. This implies that a failure report by the test system does not necessarily mean failure in the basic recognition test of Definition 3. Nonetheless, we believe that the automated test system can often point out defects in

Algorithm 2: Automated Basic Recognition Test

```
Target: $S$ with $P_L$
Input: strs: keywords or key sentences (optional)
Output: Reports for all test sentences
$Txts \leftarrow$ TestSentenceGenerater($P_L$,strs);
foreach $w \in Txts$ do
    $\mathcal{D}_w \longleftarrow$ TestAudioDataGenerater($w$);
    /** RecognitionChecker **/;
    $RTxts \longleftarrow \emptyset$;
    foreach $d \in \mathcal{D}_w$ do
        add Recognize($S$, $d$) into $RTxts$ ;
    if collectiveEqCheck($w$, $RTxts$) then
        output Test Success Report for $w$ ;
    else
        output Test Failure Report for $w$ ;
```

ASR-SUT (this will also be evaluated in the next section).

IV. Experimental Evaluation

We developed a prototype test system for the automated basic recognition test based on the ideas in Section III and performed experiments to evaluate it. Here, we present the results obtained from evaluating three matters:

BQ. Can multiple speech synthesizers generate an audio data set that can be considered equal to typical audio data?
RQ1. Can the test system detect defects in the lexicons?
RQ2. Can the system detect defects in the acoustic model?

A. *Experiment for evaluating BQ*

As discussed previously, it is necessary that **BQ** be answered affirmatively for the test system to work well. We performed an experiment using a commercially available automatic speech recognition system ASR_1 as ASR-SUT without any functional defect to evaluate this BQ. The ASR_1 is an ASR system based on deep learning [6], [18] that generally has good recognition accuracy for speech data similar to that of training data. This time, ASR_1 did not use synthesized audio data for training the acoustic model. The language model of ASR_1 is a general-purpose N-gram model for handling a wide range of speech.

In the experiment, we first generated ~100 test word sequences from both the language model and the history records of recognized texts from a call center. The generated word sequences were typical sentences from speech obtained from dialogue made during call center operations. We then synthesized an audio data set from each of the test sentences using different TTS modules $\mathsf{M}_1, \cdots, \mathsf{M}_7$. Each module was composed of multiple TTS synthesizers, each of which generated various kinds of speech sounds. Note that we can obtain various TTS synthesizers from a single TTS synthesizer by setting different values to its parameters.

M	TTS configuration of each module	$\#D_w$	WRER(%)
M1	Only female-normal	1	10.65
M2	M1 + male-normal	2	4.47
M3	M2 + female-slow + male-slow	4	4.12
M4	M3 + female-rapid + male-rapid	6	3.78
M5	M4 + all female speed-variations	15	3.09
M6	M5 + all male speed-variations	24	1.41
M7	M6 + volume power-variations	48	0.00

・M means each TTS module consisting of TTSs with different characteristics.
・Male-xx and female-xx mean male and female voices
・Normal, (rapid and slow) means normal, (rapid and slow) utterances.
・$\#D_w$ means number of synthesized audio data for each test sentence.
・WER is word error rate for speech recognition for each audio data set.

Table I
RESULTS OF EXPERIMENT FOR EVALUATING BQ.

Word	*Correct reading*	*Incorrect reading*	*Detected?*
Pittsburgh	*P IH TS B ER G*	*F IH L AX D EH L F IY AX*	Yes
July	*JH UH L AY*	*AO G AH S T*	Yes
Mike	*M AY K*	*M IH K EH*	Yes
Ticket	*T IH K IH T*	*T IH K EY T*	No
License	*L AY S IH N S*	*L AY S IH N _* (lack of *S*)	No
IEEE	*AY T R IH P AX L IY*	*AY IY IY IY* (only)	Yes
ESC	*EH S K EY P*	*IY EH S S IY* (only)	Yes

・We tested ASR1 including lexicon that changed reading of above words to incorrect ones. (only) means lexicon lacks left correct one.
・Underlined parts indicate incorrect readings.
・Detected? means whether any defect of reading was detected.

Table II
RESULTS OBTAINED FROM EXPERIMENTS FOR EVALUATING RQ1.

Finally, we had ASR_1 recognize each set of audio data and compared the resulting recognized texts with corresponding test sentences using the judgment of collectiveEqCheck with a standard synonym list. Note that all the evaluation processes can be automated.

Table I lists the average word recognition error rate *WRER* of speech recognition for the generated test word sequences. The word recognition error rate is slightly differs from standard *WER* described in Eq.(5). *WRER* can be calculated for each test word sequence w by

$$WRER(w) = E(w) \ / \ N(w), \qquad (6)$$

where $E(w)$ indicates how many words of test word sentence w have never been correctly recognized in the automated basic recognition test and $N(w)$ is the number of words in the test sentence w. For example, by using the test sentence w = "two tickets to Santa Barbara" and the set of recognition results "`two tickets to Saint Barbara`" , "`to ticket to Saint Barbara`", we have $WRER(w) = 1/5 = 0.2$, because only one word "`Saint`" has never been correctly recognized under the basic recognition test criteria.

Examples of the generated test sentences include "`This is in California so it's notfive o'clock.`" and "`Thank you for calling our help desk. How can I help you?`". For each TTS comprising a TTS module M, we used the Festival speech synthesis system [19], [20] with various settings of gender (male or female), speech rates (normal, rapid [rate=1.2], slow [rate=0.7], and *all speed variations* [rates in the range of 0.5 to 1.5 in 0.1 increments]), and volume of utterances. The *volume power variations* in the table contain all audio data obtained by applying the Hamming window process to all speech data of normal volume. More specifically, the Hamming window process with a random period from approximately 1 to 2 s was applied to the generated speech data to transform the speech power in an utterance [21].

As an example, let us consider module M_3. It is composed of four TTSs generating four speech audio data (female and male voices with normal and slow utterance speed rates) from each test sentence. This set of four audio data is recognized by ASR_1 and the resulting recognized texts are compared to the test sentence. *WRER*=4.12 is the result of these comparisons for all test statements.

As listed in Table I, the audio data generated by module M_1, which only synthesizes a female normal voice, has a *WRER* of 10.65% (statistically, one word in every ten is not correctly recognized). Therefore, we cannot consider that M_1 will generate a sufficiently typical audio data set that can be correctly recognized as a whole by ASR_1 and used as test audio data for the functional testing. However, the audio data generated by module M_7, which synthesizes 48 audio data with different characteristics, has a *WRER* of 0%. This indicates that we can answer affirmatively to **BQ**, and it may be possible to generate a speech audio data set that is correctly recognized (up to the equality of collectiveEqCheck) with very high accuracy by using only multiple TTSs with different gender, speech speed rate, and utterance volume power characteristics.

B. Experiments for evaluating RQ1 and RQ2

RQ1 and **RQ2** ask whether the test system can find functional defects in the lexicon and the acoustic model. We changed the pronunciation data on some words in the lexicon of ASR_1 to erroneous ones to evaluate RQ1 and checked whether these errors could be found with the test system. We likewise adjusted the acoustic model for question RQ2 such that the acoustic likelihoods of some phonemes were unnaturally lowered and checked whether these errors could be found by the system. We generated $\sim$100 test sentences and used M_7 in Table I for the TTS module.

Table II summarizes the defects included in the lexicon and the results obtained from the experiments for **RQ1**. The first three words were changed to incorrect readings of the same kinds of words and these defects were detected by the test system. These incorrect readings are actual defects detected in our past development of an ASR system. The lexicon may often be manually changed, and part of the lexicon of a large-scale ASR system is also often automatically generated from other sources, such as those

Results for the sentence below including *SH phonemes*: Affected word [machine: M AX *SH* IY N]	
(Test Sentence)	May I have your customer number or **machine** type and serial number please
(Male-normal)	May I have your customer number or missing type and serial number please
(Male-rapid)	May I have your customer number are nursing type and number please
(Male-slow)	May I have your customer number or who stream tarp and serial number please
(Female-normal)	May I have your customer number on the scene type and serial number please
(Female-rapid)	Well I have ya lot of customer number crunching type and serial number please
(Female-slow)	May I have your customer number Balanchine type and serial number please

Results for the sentence including *AY phonemes*: Affected word [line: L *AY* N]	
(Test Sentence)	I will have someone on the **line** with you shortly
(Male-normal)	Are you will have someone on the land with you shortly
(Male-rapid)	They will have someone on the language you shortly
(Male-slow)	But it will help someone onboard with you shortly
(Female-normal)	I will have someone on the language you shortly
(Female-rapid)	And we'll have some land and the language you're shocked me
(Female-slow)	And we'll have some Lovin hound the language you sharply

Table III
EXAMPLES OF SPEECH RECOGNITION RESULTS FOR EXPERIMENT ON RQ2.

on the Web based on various rules [22], [23]. Therefore, it is realistic to incorporate these kinds of defects. We also changed a small part of the correct reading for the next two words to one that was incorrect. The test system was unable to detect subtle errors at this level. This is because even if a slightly erroneous reading had been registered in the lexicon, the information would have been complemented by likelihoods from the acoustic and language models, and accurate recognition results would have been obtained. We think that defects at this level should be detected in the robustness test. The last two words have special technical interpretations and the incorrect lexicon was changed to remove these interpretations. These defects are detected. This type of defect is also often mixed into the lexicon.

We set abnormal values (i.e., lowered likelihoods) in the acoustic likelihoods of phonemes "SH" and "AY" in the acoustic model of ASR_1 for the experiment on question RQ2 and ran the test system against ASR_1. We used the same test sentences and TTS module M_7 as in the previous experiment. As a result, all sentences containing words for the two defective phonemes were judged as rejected by the test system, and we could detect the defects in the acoustic model. Examples of the recognition results are given in Table III, where each `(test sentence)` indicates a test sentence to be checked and `(male/female-normal/rapid/slow)` means the results of speech recognition by the defective ASR_1 of normal volume audio data generated from each of the corresponding speech synthesizers. Correctly recognized parts are underlined. In both cases, only the words "machine" and"line" affected by defective phonemes were not correctly recognized for *all* the speech data generated by male/female-normal/rapid/slow TTSs. Consequently, the test system judged that ASR_1 could not correctly recognize speech according to the two test sentences.

V. DISCUSSION

The results obtained from these experiments imply that we can answer positively to BQ, RQ1, and RQ2. That is, the experiments confirmed that we can construct an adequate test audio data generator using a TTS module and automate the basic recognition test. This section discusses the threats to the validity of these results, as well as issues concerning the actual use of the automated test system.

A. Threats to validity

Incomplete TTS-module. The primary threat to the validity of the results (especially for BQ) is whether the results can be valid for other ASR systems. This threat can be mitigated if we can construct advantageous TTS modules that answer positively to BQ for other ASR systems that are different, such as those implemented by different methods or those designed for handling different (non-English) languages. We believe we can configure such TTS modules for a wide range of ASR systems at least at a useful level. In fact, we have already configured such TTS modules for various ASR systems including ASR based on Gaussian mixture and hidden Markov models (GMM/HMM) [24], [25] and ASR for handling Japanese speech (where we used JTalk [26] as the TTS). The results from an evaluation of ASR_2, which is a commercial-grade ASR system based on GMM/HMM that handles Japanese speech, are listed in Table IV. Even in these cases, we were able to construct appropriate TTS modules simply by using TTSs in which only the gender, speech rate, and utterance volume characteristics were changed. These are known factors that greatly change speech recognition accuracy [27]. We can also use TTSs that synthesize childish or elderly voices, if necessary, and other types of voices. Because the test audio data generator and recognition checker modules are fully automated, they can work well even if it comes to using many TTSs and can handle copious amounts of test audio data.

False Positive Reports. The automated basic recognition test system may output false positive test failure reports. A false positive report means there is a test sentence such that ASR-SUT cannot correctly recognize any speech data synthesized by a TTS module for the test sentence. The false positives stem from the incompleteness of the TTS

Test Target: ASR2 (based on GMM/HMM and designed for handling Japanese)

MJ	TTS configuration	WRER(%)
J1	Male-normal	5.91
J2	J1 + male-rapid	2.76
J3	J2 + male-slow	0.78
J4	J3 + female-normal	0.12
J5	J4 + female-rapid	0.00
J6	J5 + female-slow	0.00

Evaluation: From generated about twenty test sentences, we synthesized each set of speech data by using each TTSs-module MJ (J1,…J6). Then, we had ASR2 recognize the speech data set and computed WRER for each the set of speech audio data by using collectiveEqCheck.

Table IV
ADDITIONAL RESULTS FROM EXPERIMENTS FOR EVALUATING BQ.

module used in the test system against $\mathbf{Sp}[S]$. We had no false positive reports for 100 generated test sentences in the experiments for RQ1 and RQ2. However, if we had used M_1 instead of M_7, we would have received such false positive reports. The test sentences for the false positives need to be manually tested in a robustness test. Nonetheless, we believe the test method is useful in practice, as it is possible to automatically detect many words that are never recognized due to functional defects of the ASR-SUT.

Evaluation data and its size. We used 100 sentences (containing different ~2000 words) for the evaluations. This size of data is not sufficient for a comprehensive assessment of speech recognition test for general-purpose ASRs. However, we think that our evaluations are meaningful as at least the preliminary functional evaluations of the testing method, especially for domain-specific ASRs, because there is a lot of domain-specific ASRs that handle only similarly sized words. In the speech recognition community, it is often required to perform a predetermined full-scale experiment using test data for daily new terms and non-native speakers in addition to specific benchmark data [28], [29] . Such full-scale evaluation is a future task.

B. Effects on actual ASR testing

The basic recognition test is suitable as a preliminary test step for the existing positive data recognition test on ASR systems because the new test can confirm the basic recognition abilities of ASR systems at an early test stage in a form that is separate from concerns related to the recognition accuracy and robustness of the ASR systems. It requires a constant amount of work to introduce the test system into the existing ASR test process because of three automations: (A1) generations of test sentence and test audio data are automated, (A2) test execution is fully automated, and (A3) the confirmation of test results is fully automated. Note that we need to prepare test sentences for A1, even if we do not use the basic recognition test automation. Also, the fact that additional work is constantly suppressed mainly results from A3, i.e., the expected value can be automatically determined for every given positive input in the basic recognition test.

We actually applied the automated basic recognition test system to the testing of the ASR module of a robot system for answering questions about lottery tickets. The robot was being used by a bank for facilitating lottery sales and it needed to recognize a large number of technical terms (e.g., the names of various kinds of lotteries) and typical question related to lotteries. In the development, we needed to revise and extend the lexicon and the acoustic model for handling the technical terms and typical lottery-related questions. Although it becomes an informal report for the protection of client information, the test system could detect many defects in the lexicon before we gathered test speech data. For example, the lexicon did not contain correct pronunciation data for an actor advertising the lotteries. That kind of defect was first discovered by using this test system.

VI. Related work

Various levels of testing, such as unit, integration, system, and acceptance testing [30], have been studied in the software test research field. These testing levels help detect defects in the early phases of system development. We think that the basic recognition test described in this paper introduces a new testing level to the ASR test.

Various tests and test automations for emerging software have also been studied in this field, such as GUI-based Web applications (e.g., [31]–[34]), mobile applications [35]–[38], and machine learning components [39], [40] in different eras. We need to handle and model event inputs from a graphical interface to systematically test GUI-based Web applications, as well as numerical and string values. We also need to deal with context-sensitive inputs in testing mobile applications, which change depending on where the mobile device is used or who is using the mobile application [35]. Therefore, the adoption of automated test tools is very limited in mobile application testing [36]. We are often faced with the problem that there is strictly no test oracle [2], [3] in testing statistical machine learning-based components. Reverting back to ASR testing, we need to deal with speech inputs, which are not standard in conventional software. Such input speech is essentially context-sensitive continuous data that change depending on the acoustic environment in which the speech is uttered and the person who utters the speech. Moreover, modern ASR systems are based on statistical machine learning such as deep learning. This suggests that there may be no strict test oracle in the sense that no correct recognition sentence can be obtained in advance for each input speech, and perhaps the actual output should be thought of as the correct result. We extracted the basic recognition capability of an ASR system in our approach, where the capability can be tested without any such ordinal test oracle, which is irrespective of the context in which the target speech input has been uttered.

The idea of generating test audio data using a text to speech synthesizer has already been proposed [41]. However, this article has not discussed whether the synthesized audio data is actually adequate for testing. Therefore it has not discussed the simultaneous use of multiple speech synthesizers, nor the necessity for them. As suggested by the experimental evaluations in Section IV (especially those listed in Table I), it is difficult to generate high quality speech data that is sufficient for the testing recognition capability of ASR systems with only a single speech synthesizer. Rusko et.al [42] also proposed a method to generate test speech by using speech synthesizers for testing recognition *robustness*. The method is based on the channel conversion and the addition of noise according to the target acoustic environment. However it also do not mention the quality of the synthesized speech data. Therefore, when we use such method in actual testing, we need to validate the generated speech data.

Many studies (e.g., [17], [43]) have proposed various methods of improving recognition accuracy by using recognition results from multiple speech recognition systems. In contrast, our main objective for using multiple recognition results is to test the basic recognition capability of an ASR system. We believe this approach is specific to the test viewpoint proposed in this paper. The basic recognition test can naturally be applied to ASR systems composed of multiple types of speech recognition.

There have not been many studies featuring end-to-end tests of automatic speech recognition systems as the main theme. Dookhoo [44] discussed testing speech recognition systems as his main topic and focused on the automation of various tasks in regression testing for ASR, especially when updating the language model of ASR. There are numerous opportunities in regression testing to make it easier to automate various tasks, as you can obtain the input and corresponding output used in previous tests.

VII. Conclusion

We proposed a test viewpoint for ASR systems that can check the basic capability for recognizing all target vocabulary words independently from tests on the recognition robustness. Especially from software engineering view, we formalized the basic recognition test for ASR systems and designed an automation of the testing process. We also implemented an automated test system and confirmed through several experiments that it worked well and could detect automatically functional defects of ASR systems.

Expected future work will include large scale evaluation using benchmark data adopted in the speech recognition community as stated in Section V. We are also considering a method of systematic high coverage test sentence generation from large and stochastic language models. Systematic speech data generation to test recognition robustness is also a future task. The language model has information on the ease of connecting each word, i.e., information on parts that are difficult to correctly recognize. Thus, it may be possible to automatically generate efficient speech data to test robustness by using this information.

References

[1] J.-C. Junqua and J.-P. Haton, *Robustness in Automatic Speech Recognition: Fundamentals and Applications*. Norwell, MA, USA: Kluwer Academic Publishers, 1995.

[2] E. J. Weyuker, "On testing non-testable programs," *The Computer Journal*, vol. 25, no. 4, pp. 465–470, 1982.

[3] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE transactions on software engineering*, vol. 41, no. 5, pp. 507–525, 2015.

[4] D. Yu and L. Deng, *Automatic Speech Recognition: A Deep Learning Approach*. Springer Publishing Company, Incorporated, 2014.

[5] X. Huang, Y. Ariki, and M. Jack, *Hidden Markov Models for Speech Recognition*. New York, NY, USA: Columbia University Press, 1990.

[6] T. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013)*, March 2013.

[7] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002. [Online]. Available: http://dx.doi.org/10.1006/csla.2001.0184

[8] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Comput. Linguist.*, vol. 18, no. 4, pp. 467–479, Dec. 1992. [Online]. Available: http://dl.acm.org/citation.cfm?id=176313.176316

[9] The International Computer Science Institute, Speech Group, "sclite - score speech recognition system output," http://www1.icsi.berkeley.edu/Speech/docs/sctk-1.2/sclite.

[10] H. Sakoe, "Readings in speech recognition," A. Waibel and K.-F. Lee, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, ch. Two-level DP-matching – A Dynamic Programming-based Pattern Matching Algorithm for Connected Word Recognition, pp. 180–187. [Online]. Available: http://dl.acm.org/citation.cfm?id=108235.108246

[11] J. Holmes and W. Holmes, *Speech Synthesis and Recognition*, 2nd ed. Bristol, PA, USA: Taylor & Francis, Inc., 2002.

[12] J. E. Hopcroft and J. D. Ullman, *Introduction To Automata Theory, Languages, And Computation*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1990.

[13] I. Langkilde-Geary, "An empirical verification of coverage and correctness for a general-purpose sentence generator," in *Proceedings of the International Natural Language Generation Conference*, 2002. [Online]. Available: http://www.aclweb.org/anthology/W02-2103

[14] B. Pang, K. Knight, and D. Marcu, "Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, ser. NAACL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 102–109. [Online]. Available: http://dx.doi.org/10.3115/1073445.1073469

[15] O. Ichikawa, S. J. Rennie, T. Fukuda, and M. Nishimura, "Channel-mapping for speech corpus recycling," in *Proc. of 2013 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013)*, IEEE, 2013, pp. 7160–7164.

[16] T. Fukuda, O. Ichikawa, M. Nishimura, S. J. Rennie, and V. Goel, "Regularized feature-space discriminative adaptation for robust asr," in *Proc. of 15th Annual Conference on the International Speech Communication Association (Interspeech)*, 2014.

[17] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus in speech recognition: word error minimization and other applications of confusion networks," *Computer Speech and Language*, vol. 14, 2000.

[18] T. Fukuda, R. Tachibana, U. Chaudhari, B. Ramabhadran, and P. Zhan, "Constructing ensembles of dissimilar acoustic models using hidden attributes of training data," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2012)*, March 2012.

[19] P. Taylor, A. W. Black, and R. Caley, "The architecture of the festival speech synthesis system," in *IN THE THIRD ESCA WORKSHOP IN SPEECH SYNTHESIS*, 1998, pp. 147–151.

[20] The Centre for Speech Technology Research,The University of Edinburgh, "The festival speech synthesis system," http://www.cstr.ed.ac.uk/projects/festival/.

[21] E. Vincent, S. Watanabe, A. A. Nugraha, J. Barker, and R. Marxer, "An analysis of environment, microphone and data simulation mismatches in robust speech recognition," *Computer Speech and Language*, Nov. 2016. [Online]. Available: https://hal.inria.fr/hal-01399180

[22] S. F. Chen *et al.*, "Conditional and joint models for grapheme-to-phoneme conversion." in *INTERSPEECH*, 2003.

[23] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Commun.*, vol. 50, no. 5, pp. 434–451, May 2008. [Online]. Available: http://dx.doi.org/10.1016/j.specom.2008.01.002

[24] J. Bilmes, "A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," Tech. Rep., 1998.

[25] M. Gales and S. Young, "The application of hidden markov models in speech recognition," *Found. Trends Signal Process.*, vol. 1, no. 3, pp. 195–304, Jan. 2007. [Online]. Available: http://dx.doi.org/10.1561/2000000004

[26] "Open jtalk," http://open-jtalk.sourceforge.net/.

[27] F. Beaufays, V. Vanhoucke, and B. Strope, "Unsupervised discovery and training of maximally dissimilar cluster models," in *Proc Interspeech*, 2010.

[28] J. Godfrey, E. Holliman, and J. McDaniel, "Switchboard: telephone speech corpus for research and development . acoustics,," in *IEEE International Conference on Speech, and Signal Processing, ICASSP-92*, vol. 1, 1992, pp. 517–520.

[29] K. Maekawa, H. Koiso, S. Furui, and H. Isahara, "Spontaneous speech corpus of japanese," in *Proceedings of the Second International Conference on Language Resources and Evaluation, LREC 2000, 31 May - June 2, 2000, Athens, Greece*, 2000. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2000/pdf/262.pdf

[30] I. C. Society, P. Bourque, and R. E. Fairley, *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*, 3rd ed. Los Alamitos, CA, USA: IEEE Computer Society Press, 2014.

[31] F. Ricca and P. Tonella, "Analysis and testing of web applications," in *Proceedings of the 23rd International Conference on Software Engineering*, ser. ICSE '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 25–34. [Online]. Available: http://dl.acm.org/citation.cfm?id=381473.381476

[32] A. M. Memon, "Gui testing: Pitfalls and process," *IEEE Computer*, vol. 35, no. 8, pp. 87–88, 2002.

[33] A. M. Memon, I. Banerjee, and A. Nagarajan, "Gui ripping: Reverse engineering of graphical user interfaces for testing." in *WCRE*, vol. 3, 2003, p. 260.

[34] M. Vieira, J. Leduc, B. Hasling, R. Subramanyan, and J. Kazmeier, "Automation of gui testing using a model-driven approach," in *Proceedings of the 2006 international workshop on Automation of software test*. ACM, 2006, pp. 9–14.

[35] H. Muccini, A. Di Francesco, and P. Esposito, "Software testing of mobile applications: Challenges and future research directions," in *Proceedings of the 7th International Workshop on Automation of Software Test*, ser. AST '12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 29–35. [Online]. Available: http://dl.acm.org/citation.cfm?id=2663608.2663615

[36] V. L. L. Dantas, F. G. Marinho, A. L. da Costa, and R. M. Andrade, "Testing requirements for mobile applications," in *Computer and Information Sciences, 2009. ISCIS 2009. 24th International Symposium on*. IEEE, 2009, pp. 555–560.

[37] N. Z. binti Ayob, A. R. C. Hussin, and H. M. Dahlan, "Three layers design guideline for mobile application," in *Information Management and Engineering, 2009. ICIME'09. International Conference on*. IEEE, 2009, pp. 427–431.

[38] Y. Hu, I. Neamtiu, and A. Alavi, "Automatically verifying and reproducing event-based races in android apps," in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, ser. ISSTA 2016. New York, NY, USA: ACM, 2016, pp. 377–388. [Online]. Available: http://doi.acm.org/10.1145/2931037.2931069

[39] X. Xie, J. W. K. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Testing and validating machine learning classifiers by metamorphic testing," *J. Syst. Softw.*, vol. 84, no. 4, pp. 544–558, Apr. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2010.11.920

[40] C. Murphy, G. Kaiser, and M. Arias, "An approach to software testing of machine learning applications," in *Proceedings of 19th SEKE*, 2007, pp. 167–172.

[41] H. Crepy, J. Kusnitz, and B. Lewis, "Testing speech recognition systems using test data generated by text-to-speech conversion," US Patent No. 6,622,121 B1, Sep 2003.

[42] M. Rusko, M. Trnka, S. Darjaa, R. Sabo, S. B. Juraj Palfy, and M. Ritomsky, "Test signals generator for asr under noisy and reverberant conditions using expressive tts," in *FORUM ACUSTICUM*, European Acoustics Association, 2013.

[43] J. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover)," in *IEEE Workshop on Automatic Speech Recognition and Understanding*, 1997.

[44] R. A. Dookhoo, "Automated regression testing approach to expansion and refinement of speech recognition grammars," Master Thesis,University of Central Florida, University of Central Florida, 2008.