SPRINGER BRIEFS IN COMPUTER SCIENCE

Yutaka Kidawara
Eiichiro Sumita
Hisashi Kawai *Editors*

Speech-to-Speech Translation

Springer

SpringerBriefs in Computer Science

SpringerBriefs present concise summaries of cutting-edge research and practical applications across a wide spectrum of fields. Featuring compact volumes of 50 to 125 pages, the series covers a range of content from professional to academic.

Typical topics might include:

- A timely report of state-of-the art analytical techniques
- A bridge between new research results, as published in journal articles, and a contextual literature review
- A snapshot of a hot or emerging topic
- An in-depth case study or clinical example
- A presentation of core concepts that students must understand in order to make independent contributions

Briefs allow authors to present their ideas and readers to absorb them with minimal time investment. Briefs will be published as part of Springer's eBook collection, with millions of users worldwide. In addition, Briefs will be available for individual print and electronic purchase. Briefs are characterized by fast, global electronic dissemination, standard publishing contracts, easy-to-use manuscript preparation and formatting guidelines, and expedited production schedules. We aim for publication 8–12 weeks after acceptance. Both solicited and unsolicited manuscripts are considered for publication in this series.

More information about this series at http://www.springer.com/series/10028

Yutaka Kidawara · Eiichiro Sumita ·
Hisashi Kawai
Editors

Speech-to-Speech Translation

Springer

Editors
Yutaka Kidawara
Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology
Kyoto, Japan

Hisashi Kawai
Advanced Speech Technology Laboratory, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology
Kyoto, Japan

Eiichiro Sumita
Advanced Translation Technology Laboratory, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology
Kyoto, Japan

ISSN 2191-5768 ISSN 2191-5776 (electronic)
SpringerBriefs in Computer Science
ISBN 978-981-15-0594-2 ISBN 978-981-15-0595-9 (eBook)
https://doi.org/10.1007/978-981-15-0595-9

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Preface

"Language barrier" is one of the biggest challenges that humans face in communication. In the Old Testament, it is said that God created different languages when humans acted in defiance and attempted to build the Tower of Babel against His wishes. Whether this is true or not, it is an unquestionable fact that smooth communication has been hindered because of the different words we speak. Speech translation device—often makes its appearance in science fiction—which immediately translates spoken words into different languages has been the target of research and development in Japan since 1986. After some 30 years of constant effort, this dream device has been made available for practical use in certain fields and under certain conditions.

The history of research and development on machine translation can be traced back to the 1940s, but it wasn't until the 1983 World Telecommunication Exhibition (TELECOM 83) that multilingual speech translation technology—speech recognition and speech synthesis combined with machine translation—really caught people's attention. In 1986, the Advanced Telecommunications Research Institute International (ATR) was established in Japan, followed by the launch of a research project on speech-to-speech translation. Researchers from around the world gathered to conduct research and development under this project. In the 1990s, machine translation was a dictionary- and rule-based technology, but by the 2000s, alongside the diffusion of the Internet and the Web, the focus on research and development shifted to a bilingual corpus-based statistical approach, which took advantage of not having to consider the rules such as grammar.

In July 2010, the National Institute of Information and Communications Technology (NICT) initiated the world's first field experiment of a network-based multilingual speech-to-speech translation system using a smartphone application. To support further languages, NICT then led the establishment of an international consortium to initiate a global field experiment in cooperation with as many as 32 research entities of the world who had been individually engaged in developing speech translation technologies for their own languages. In April 2014, a governmental project "Global Communication Plan (GCP)" was launched in Japan to promote research and development on multilingual speech translation technology

and its implementation to fields such as travel, medical care, disaster prevention, and daily living, with the aim to eliminate the language barriers for foreign nationals and inbound tourists. With NICT taking the initiative, the industry, academia, and government together are engaged in research and development of a high-precision multilingual speech translation technology, namely by applying deep-learning approaches. NICT's speech translation application "VoiceTra" has been released as part of the field experiment under the GCP and serves as the key platform for its promotion. Thanks to the efforts from the initial field experiment and the achievements from the international collaboration, the application is currently capable of handling speech translation between as many as 16 languages and further improvements are being made, especially for the 10 most frequently used languages[1] in Japan that have been set forth as the target under the GCP.

After Google announced the development of the Google Neural Machine Translation in September 2016, a significant paradigm shift from statistical to neural machine translation has been taking place. The latest version of VoiceTra has also applied neural machine translation between Japanese and English and achieved dramatic improvements in accuracy. Let us look at a couple of comparison examples of different translation applications:

(1) The English translation results to the Japanese input "今日は全国的に雨の予報です" were:

- "Today, it is forecasted to rain all over the country" (by VoiceTra);
- "Today's rain forecast nationwide" (by App X);
- "It is forecast of rain nationwide today" (by App Y); and

(2) The English translation results to the Japanese input "昨今のニューラル翻訳システムの性能は目を見張るものがある" were:

- "The performance of the recent neural translation system is eye-catching" (by VoiceTra);
- "The performance of recent neural translation systems is spectacular" (by App X);
- "The performance of recent neural translation systems is remarkable" (by App Y).

As you may see, the difference in the level of performance between different translation systems for general use is hard to tell and choosing one over another is a difficult task. However, while further evaluation is required, we may see from a few more examples below that systems targeted for specific purposes, i.e., VoiceTra, as mentioned earlier, achieve better results in terms of accuracy within the targeted domain due to the sufficient number of corpora and dictionaries prepared for the respective domains.

[1]Japanese, English, Chinese, Korean, Thai, French, Indonesian, Vietnamese, Spanish, and Myanmar.

(1) [Domain: Daily living] The English translation results to the Japanese input "トイレが流れません" were:

- "The toilet doesn't flush" (by VoiceTra, Evaluation[2]: A);
- "The toilet does not flow" (by App X and Y, Evaluation: B);

(2) [Domain: Travel] the English translation results to the Japanese input "富士山五合目まではバスで行けます" were:

- "You can go to the fifth station of Mount Fuji by bus" (by VoiceTra, Evaluation A);
- "It is possible to go by bus to Mt. Fuji" (by App X, Evaluation: C); and
- "You can go by bus to the 5th of Mt. Fuji" (by App Y, Evaluation: B); and lastly,

(3) [Domain: Medical care] the English translation results to the Japanese input "認知症というと、治らないものばかりだと思っていました" were:

- "I thought dementia is not curable" (by VoiceTra, Evaluation: A);
- "When thinking of dementia, I thought that only things did not go away" (by App X, Evaluation: C); and
- "I thought that dementia was the only thing that was not cured" (by App Y, Evaluation: C).

This book explores the fundamentals of the research and development on multilingual speech-to-speech translation technology and its social implementation process which have been conducted as part of the GCP. The practicability of such technology is rapidly increasing due to the latest developments in deep-learning algorithms. Multilingual speech translation applications for smartphones are very handy, however, concerns do exist on the interactivity of the operation that may keep smooth communication from being ensured in real use; therefore, the development of translation devices for specific purposes is also ongoing. Namely, easy-to-use, high-precision translation services that are distinctive from those for smartphones and tablets are being developed by industries, and the final sections of this book will present one example of such practical technology specifically designed for hospitals. In such way, the GCP is an unprecedented type of scheme allowing open innovation to utilize multilingual translation technology in every corner of society.

Yutaka Kidawara
Director General
Advanced Speech Translation Research and
Development Promotion Center (ASTREC), NICT Kyoto Japan

[2]The samples were evaluated on a scale of A (perfectly accurate) – D (inaccurate).

Acknowledgements We would like to express our sincere gratitude to the researchers at ASTREC, NICT for their cooperation in writing this book and to Kensuke Ishii for his tremendous editorial support. We would also like to express our appreciation to the Ministry of Internal Affairs and Communications for their cooperation in promoting the GCP. Finally, we would like to thank all the participants of the Council for Global Communication Development Promotion and everyone who has taken part in the ongoing field experiments.

Contents

Contributors

Yutaka Ashikari System Development Office, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology, Kyoto, Japan

Masakiyo Fujimoto Advanced Speech Technology Laboratory, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology, Kyoto, Japan

Shoji Harada Fujitsu Laboratories Ltd., Kanagawa, Japan

Kenji Imamura Advanced Translation Technology Laboratory, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology, Kyoto, Japan

Hiroaki Kato Advanced Speech Technology Laboratory, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology, Kyoto, Japan

Hisashi Kawai Advanced Speech Technology Laboratory, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology, Kyoto, Japan

Tasuku Kitade Biometrics Research Laboratories, NEC Corporation, Kanagawa, Japan

Sheng Li Advanced Speech Technology Laboratory, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology, Kyoto, Japan

Xugang Lu Advanced Speech Technology Laboratory, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology, Kyoto, Japan

Jinfu Ni Advanced Speech Technology Laboratory, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology, Kyoto, Japan

Takuma Okamoto Advanced Speech Technology Laboratory, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology, Kyoto, Japan

Yoshinori Shiga Advanced Speech Technology Laboratory, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology, Kyoto, Japan

Fumiaki Sugaya MINDWORD, Inc., Tokyo, Japan

Eiichiro Sumita Advanced Translation Technology Laboratory, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology, Kyoto, Japan

Kentaro Tachibana AI System Department, AI Unit, DeNA Co., Ltd., Tokyo, Japan

Keiji Yasuda Data Science Center, Nara Institute of Science and Technology, Nara, Japan

Chapter 1
Multilingualization of Speech Processing

Hiroaki Kato, Shoji Harada, Tasuku Kitade, and Yoshinori Shiga

Abstract Speech-to-speech translation is a technology that connects people of different languages together and its multilingualization dramatically expands the circle of people connected. "Population" in Table 1.1a shows the potential number of people who can be part of the circle, when the corresponding language benefits from the technology. However, the same table also tells us that the languages of the world are incredibly diverse, and therefore multilingualization is not an easy task. Nevertheless, methods of processing speech sounds have been devised and developed uniformly regardless of language differences. What made this possible, is the wide commonality across languages due to the nature of language—it is a spontaneous tool created for the single purpose of mutual communication between humans who basically share the same biological hardware. This chapter will describe the multilingualization of automatic speech recognition (ASR) and text-to-speech synthesis (TTS); the two speech-related components of the three that constitute the speech-to-speech translation technology.

S. Harada and T. Kitade belonged to NICT at the time of writing.

H. Kato (✉) · Y. Shiga
Advanced Speech Technology Laboratory, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology, Kyoto, Japan
e-mail: kato.hiroaki@nict.go.jp

Y. Shiga
e-mail: yoshinori.shiga@nict.go.jp

S. Harada
Fujitsu Laboratories Ltd., Kanagawa, Japan
e-mail: show.harada@fujitsu.com

T. Kitade
Biometrics Research Laboratories, NEC Corporation, Kanagawa, Japan
e-mail: t-kitade@ay.jp.nec.com

Y. Kidawara et al. (eds.), *Speech-to-Speech Translation*, SpringerBriefs in Computer Science, https://doi.org/10.1007/978-981-15-0595-9_1

Table 1.1 Language characteristics

(a)

Language name (2nd row: in the orthography)	Population (million people)[1]	Language family	Morphological type	Word order[2]	Number of phonemes[3]
Japanese にほん語	127 (L1)	Japonic	Agglutinative	SOV	5 v, 14 c 0 t
English English	360 (L1) 1860 (+L2)	Germanic Indo-European	Fusional	SVO	11 v, 24 c 0 t
Chinese 汉语	1200 ~ (L1)	Sinitic Sino-Tibetan	Isolating	SVO	6 v, 19 c 4 t
Korean 한국어	77 (L1)	Koreanic	Agglutinative	SOV	9 v, 19 c 0 t
Thai ภาษาไทย	60 (L1)	Tai-Kadai	Isolating	SVO	9 v, 21 c 5 t
Vietnamese Tiếng Việt	75 (L1)	Mon-Khmer Austroasiatic	Isolating	SVO (OSV)	11 v, 19 c 6 t
Indonesian Bahasa Indonesia	23 (L1) 163 (+L2)	Austronesian	Agglutinative	SVO	6 v, 18 c 0 t
Myanmar မြန်မာဘာသာ	33 (L1) 44 (+L2)	Tibeto-Burman Sino-Tibetan	Agglutinative	SOV	8 v, 23 c 3 t
French Français	75 (L1) 338 (+L2)	Italic Indo-European	Fusional	SVO	11 v, 20 c 0 t
Spanish Idioma español	438 (L1) 528 (+L2)	Italic Indo-European	Fusional	SVO	5 v, 17 c 0 t
Portuguese Português	220 (L1) 250 (+L2)	Italic Indo-European	Fusional	SVO	8 v, 19 c 0 t
Russian Русский язык	150 (L1) 260 (+L2)	Slavic Indo-European	Fusional	SVO	10 v, 32 c 0 t
Arabic اللغة العربية	340 (L1)	Semitic Afro-Asiatic	Fusional	VSO (SVO)	3 v, 28 c 0 t
Khmer ភាសាខ្មែរ	16 (L1)	Mon-Khmer Austroasiatic	Isolating	SVO	9 v, 21 c 0 t

(b)

Lang. code[4]	Writing system	Number of graphemes[5]	Writing direction	Word separator	Numerals[6]
ja, jpn	Chinese logograms + Kana phonograms	2998 lg[7] + 96 ph	Left-to-right up-to-down	NA	〇一二三四五六七八九
en, eng	Alphabetical phonograms	26 ph (Latin)	Left-to-right	White spaces	NA
zh, zho	Chinese logograms	8105 lg[7]	Left-to-right up-to-down	NA	〇一二三四五六七八九
ko, kor	Hangul phonograms	24 ph (Hangul)	Left-to-right up-to-down	White spaces[8]	NA
th, tha	Brahmic phonograms	42 ph (Thai)	Left-to-right	NA	๐๑๒๓๔๕๖๗๘๙
vi, vie	Alphabetical phonograms	29 ph (Latin)	Left-to-right	White spaces	NA

(continued)

Table 1.1 (continued)

(b)					
Lang. code[4]	Writing system	Number of graphemes[5]	Writing direction	Word separator	Numerals[6]
id, ind	Alphabetical phonograms	26 ph (Latin)	Left-to-right	White spaces	NA
my, mya	Brahmic phonograms	33 ph (Myanmar)	Left-to-right	NA	၀၁၂၃၄၅၆၇၈၉
fr, fra	Alphabetical phonograms	28 ph (Latin)	Left-to-right	White spaces	NA
es, spa	Alphabetical phonograms	27 ph (Latin)	Left-to-right	White spaces	NA
pt, por	Alphabetical phonograms	26 ph (Latin)	Left-to-right	White spaces	NA
ru, rus	Alphabetical phonograms	33 ph (Cyrillic)	Left-to-right	White spaces	NA
ar, ara	Arabic phonograms	28 ph (Arabic)	Right-to-left	NA	٠١٢٣٤٥٦٧٨٩
km, khm	Brahmic phonograms	33 ph (Khmer)	Left-to-right	NA	០១២៣៤៥៦៧៨៩

[1]L1: Native language population. +L2: Native and second-language population
[2]Dominant word order of subject, object, and verb
[3]Vowels (v), consonants (c), and tones (t). Long vowels, diphthongs, nasal vowels and medial vowels and consonants are excluded
[4]Language code by ISO639-1 (2 letters) and ISO639-2T (3 letters)
[5]Logograms (lg) and phonograms (ph). Diacritics are excluded
[6]Figures corresponding to 0–9, used except for Arabic numerals
[7]Official regular-use character set in each country
[8]Occasionally inserted between two words or word sequences

1.1 Basic Framework of Speech Processing and Its Multilingualization

This section describes a brief outline of the basic framework of speech processing and its multilingualization. Tables 1.1a, b list the characteristics of the languages that the authors' R&D group has dealt with in the past. All text and speech samples exemplified in this section are taken from the languages listed on the tables.

1.1.1 Diversity and Commonality in Spoken Languages

1. Structure of Language

(a) Phoneme and Phone

Phonemes and phones are distinctly different terms, although they are not always used properly. Phoneme is the minimal unit of speech sounds that allow us to distinguish words from one another. Since words are language dependent, the definition of "phoneme" differs between languages. The set of phonemes is fixed in a language or dialect and it is typically divided into vowels and consonants. On the other hand, "phone" is defined as the smallest perceptible discrete segment of a speech sound. Practically, "perceptible" would depend upon whether a trained expert can distinguish or judge that one is different from the other. The set of phones is fixed regardless of languages, but its size greatly fluctuates depending on how precisely each element is distinguished (see [1] for details). A phoneme may correspond to one or more phones. For example, the phoneme of the second consonant in the English word "water" is always fixed as "t", while phoneme "t" can either be an alveolar plosive consonant phone "t" or an alveolar tap consonant phone "r". Different phones included in the same phoneme are called allophones. Lexical tones are regarded as phonemes as they contribute to word distinction. For example, Vietnamese has six lexical tones or tonal phonemes. Table 1.1a lists the numbers of phonemes, i.e., vowels, consonants, and tones, in each language.

(b) Morpheme and Word

The minimal distinctive unit of grammar is called morpheme. Morpheme is also the smallest unit that contains meaning. A word consists of one or more morphemes and is intuitively known as the unit of expression, especially in languages whose writing system has a word separator (see Table 1.1b). Its grammatical function is called part-of-speech (POS). According to POS, all words are classified into content words having unique meanings such as nouns, verbs, adjectives, and function words having primarily grammatical functions such as prepositions and conjunctions. In speech processing, "word" is an indispensable concept as it serves as the basic unit of language models (Sect. 1.2.3), the heading of pronunciation dictionaries, and the metric unit in evaluating the performance of a system. Note, however, that the linguistic definition of "word" is in fact ambiguous, and therefore it is flexibly or conveniently defined in speech processing, which may in cases disagree with the native speakers' intuitions.

(c) Writing System and Grapheme

The writing system may be the most impressive example that shows the diversity of languages (see the 1st column of Table 1.1a). Speech processing must face the diversity in the writing system as its mission is to accomplish mutual conversion between speech and text. The conventional writing system of a language is called orthography, and the minimal contrastive unit in the system is called grapheme.

In modern natural languages, most graphemes can be classified into two types: logograms[1] such as kanjis, and phonograms[2] such as alphabets. Table 1.1b lists the writing systems and the number of graphemes in each language.

In logograms, each character has its own meaning. For example, the character "耳" originates from the shape of a human ear and means "ear." As different languages have different sounds in expressing "ear," the reading differs in each language; it is read as "er" in Chinese, and "mimi" in Japanese (original lexicon).

In phonograms, each letter has no unique meaning and it only represents a sound. However, languages whose letters and sounds perfectly correspond one-to-one with each other are extremely rare to find and generally have many irregularities and exceptions. Among the languages that adopt the phonographical writing system as shown in Table 1.1b, Indonesian and Vietnamese relatively maintain a regular correspondence between letters and sounds, while English does not; seeing that "write" and "right" are pronounced the same. The incongruity between spelling and pronunciation in most cases is caused by the chronological change of pronunciation. The modern orthography of Indonesian and Vietnamese has a relatively short history. Since speech processing uses phonemes as the basic unit, conversion from a grapheme to the corresponding phoneme notation (this is called G2P) is essential regardless of the writing system.

2. Acoustic Features of Speech

Speech processing in speech-to-speech translation basically deals with common acoustic features that are independent of languages, because it only targets the sounds that humans can produce and uses only the features that humans can exploit through their auditory system. The three independent attributes that humans can perceptually extract from a sound, i.e., the three attributes of tone, are loudness, pitch, and timbre. Their corresponding physical quantities are power, fundamental frequency (fo), and spectrum. ASR and TTS stand primarily on these acoustic features.

The spectrum, typically the power spectrum density, is generally used as its contour information (e.g. cepstral coefficients) by smoothing out its fine structure. The spectral contour closely approximates the resonance characteristics of the

[1]Kanji is assumed to be the only logographic system that occupies the status of the standard grapheme in modern languages. It has been popular in East Asia; in addition to modern Chinese and Japanese, it was also the standard grapheme in Korean (still used occasionally) and Vietnamese in the past.

[2]Three distinct major groups are found in phonograms all of which are said to have direct or indirect roots in Mesopotamia. First, alphabetical systems, which spread out to the West, mainly developed in Europe and then to the world, assign separate characters (not diacritic marks) for both vowels and consonants, exemplified by Cyrillic and Latin scripts. Next, the Brahmic family, which first prevailed in India, and onto other parts of South Asia and Southeast Asia, basically has only consonant characters with designation of vowels and tones (if any) as diacritic marks, including Khmer, Thai and Myanmar scripts. The last group, which remained in the Middle Eastern region, is represented by the Arabic script and is basically composed of only consonant characters, and the designation of vowels is optional.

acoustic cavities involved in speech production, i.e., the vocal tract, and therefore reflects the geometrical status of the vocal tract at a certain moment in speech activity. Thus, it tells us, for example, where the tongue tip was in the oral cavity, how it moved, if the sound had passed through the nasal cavity, and so on. These are the exact information which characterize the vowels and consonants, excluding voiced/unvoiced contrasts.

The fo indicates the presence or absence of the vibration of the vocal folds and its frequency. The sound in which a clear fo cannot be observed is called unvoiced and that with a clear fo is called voiced; this contrast is typically found between "p, t, k, s, f" and "b, d, g, z, v". In addition, the temporal pattern of fo is a major clue in distinguishing lexical tones. A well-known role of fo, aside from the above lexical information, is the expression of speech called intonation. In languages in which literal information does not distinguish declarative and yes-no interrogative sentences such as Portuguese, the meaning may not be correctly recognized without fo.

ASR uses the spectrum and fo (or the quantities derived therefrom) as the main acoustic features. On the other hand, power, fo, and spectrum roughly correspond to the basic elements that can be controlled pseudo-independently in the human speech production process, which are: energy supply from the lung, vocal-fold vibration, and vocal-tract shape. Therefore, by independently manipulating these acoustic features, TTS can generate a flexible and naturally sounding speech.

1.1.2 Challenges Toward Multilingualization

1. Speech Corpus

The system development of ASR and TTS is equivalent to teaching machines about the correspondence between speech and text. Perhaps the teaching methodology can be shared with different languages, but the teaching materials must be different. Thus, it is safe to assume that the creation of teaching materials is the most fundamental task in multilingualization. Then, how many and what kind of materials are necessary?

In theory, the least amount of material necessary would be the set of phonemes, each of which should be accompanied by its corresponding sound, in order to distinguish the words in a language. Guessing from Table 1.1a, the total number of phonemes per language would not exceed 50. Since the duration of a single phoneme falls within the range of approximately 5–200 ms—say an average of 100 ms—the total duration would be $100 \times 50 = 5000$ ms. This is only 5 s—unfortunately this does not work, because a phoneme does not correspond to a single sound but to a countless number of sounds. The same phoneme can be differently uttered depending on the phonemic context (Sect. 1.2.2), and many other contextual factors (Table 3.1), in addition to the difference in speakers. Therefore, a certain amount of sentence speech with variations from real environments and the corresponding orthographical text material are required. This is called a speech corpus.

The requirements for speech corpora are greatly different between ASR and TTS. A general-purpose ASR needs to recognize speech of unspecified users under various environments. Therefore, a large number of speakers are required with well-balanced genders, ages, and places of birth. The intensity and contents of background noise should be ranged as much as possible. The speaking style can be read, spontaneous or both. The total duration per person can be short.

In contrast, a general-purpose TTS should simply provide a single speaker's voice. Therefore, only one speaker is required (or a male and a female, if possible). It is ideal if he/she speaks fluently and naturally with a clear articulation without any trace of accent (dialectal or sociolectal). The recording environment must be extremely quiet with minimum reverberation. The speaking style and speed must be kept constant as much as possible throughout the whole recording session. The total duration per person must be far longer than that of ASR.

According to the authors' experiences, even when limiting the domain to a certain extent, the amount of speech corpora necessary to develop a minimum practical system seems to be a bit longer than 5 s; a rough estimation is about 5 h per speaker for TTS, or about 500 h (e.g., 3000 speakers) for ASR.

2. Construction of Speech Corpora

The procedure of creating a speech corpus is fundamentally independent of languages. Taking the case of ASR as an example, it typically requires four steps: design, recording, transcription, and inspection. With a read aloud speaking style, transcription is basically unnecessary. This is often the case for TTS.

Design: The process of determining the specifications such as the overall size, domain, speaking style, the speaker and dialect conditions, background noise, recording device and digitizing format, etc. The designing of inspection methods should not be forgotten. It is better to consider the consistency with the translation component in determining the domain. If the output of the recognition component contains an unknown word for the translation component, the translation will fail. Using a part of the text corpora employed to train the translation component as the manuscript of speech corpus is a promising solution, especially when the amount of collectable speech is limited.

Recording: The process of collecting utterances from native speakers. When the speaking style is spontaneous, ideas for drawing out a great variety of speech within the limited time are essential. For example, prompting interactive dialogues under preset scenes such as shopping etc., or prompting the speakers to answer predetermined questions, and so on.

Transcription: The process of transcribing the recorded utterances. This step is usually most time-consuming. Even expert transcribers typically take more than ten times than the actual time (playback time) to transcribe regular-speed spontaneous utterances.

Inspection: The process of checking and correcting the corpus. This step must not be skipped as errors inevitably occur in human-related performance, i.e., reading/speaking, hearing, and writing. Defects directly lead to the deterioration of

the recognition performance. The inspection process can be divided into automatic and manual ones.

The construction of speech corpora prominently takes the highest human and temporal costs among all processes of multilingualization in statistical speech processing technology, which is presently the mainstream. However, the cost can be dramatically reduced once a prototypical ASR system of the target language is developed, even if it were a very primitive one. For example, distributing a prototype system as an (free) application to native speakers for trial use, and collecting the users' utterances, may substitute the **recording** step; the users' consents are indeed necessary. The **transcription** step can be done based on the recognition results of the prototype system, which is probably more efficient than transcribing them from scratch. As for the **inspection** step, the prototype system can be used to automatically estimate whether a given manual transcription requires a careful inspection. If the manual transcription of an utterance largely disagrees with the ASR result of the same utterance, the transcription is assumed to contain errors.

3. Participation of Native Speakers

As mentioned above and in the remaining part, the framework and procedures of speech processing are basically common regardless of languages. Therefore, the multilingualization task does not require native speakers of the target language on a constant basis, except for corpus speakers and transcribers. However, there are also a few other aspects that definitely require the native speakers' participation, as listed below.

In constructing a speech corpus, it is strongly required for a native speaker who is aware of the corpus specifications to attend the recording as the director or the director's advisor. This prevents the recording from continuing with under-qualified utterances by checking the naturalness and acceptability as well as the correctness of the speech contents on site. The participation of native speakers is also indispensable in the inspection step; non-native speakers are not capable of judging the eligibility of an utterance or transcription based on subtle information such as accentedness and the fluctuation of spelling.

Apart from the construction of speech corpora, a native speaker's advice is essential for the development of automatic text processors such as word segmenter, G2P, and text normalizer, which are utilized to improve the efficiency of post-processing of the corpora introduced in the following sections. For example, even in a language whose orthography specifies a word boundary, it is not always optimal to use apparent words—separated by orthographical word delimiters—as the unit in word segmentation. Such problems can be fundamentally solved only by close, preferably, face-to-face communications between non-native developers and native speakers.

Once an appropriate speech corpus is constructed through the above procedures, grab useful tools such as word segmenter and G2P, and you are ready to step into the following Sects. 1.2 (ASR) and 1.3 (TTS), which will further guide you toward the multilingualization of speech processing.

1.2 Multilingualization of Speech Recognition

The automatic conversion from speech to text is called "automatic speech recognition (ASR)." ASR however, is not a uni-directional process from sounds to letters, but rather bi-directional. It estimates from the acoustic features of an input sound to determine which of the phonemes in a certain language it corresponds to, while also estimating from text information to determine which words of the language the string of sounds is most likely to match with. The statistical model used in the former process is called the acoustic model and the one used in the latter is called the language model. Both are language-specific, and therefore must be constructed for each language. This section first explains the basic framework of ASR, and then outlines the multilingualization process of the acoustic model and language model, respectively.

1.2.1 Basic Framework of Speech Recognition

In general, as shown in Fig. 1.1, speech recognition is performed using an acoustic model which associates acoustic feature sequences with phoneme sequences, and a language model which associates phoneme sequences with word sequences, and by converting speech into a sentence (word sequences) via a matching process called decoding. In the process, the decoder derives the most suitable word sequence by weighing various phoneme sequence candidates from the acoustic model and word sequence candidates from the language model, based on language-independent common searching procedures as further explained in Sects. 1.2.2 and 1.2.3. The acoustic and language models that the decoder refers to are language dependent. However, as explained in the following sections, the fundamental structure of both models is rather common among different languages, and therefore speech recognition for various languages can use a common framework.

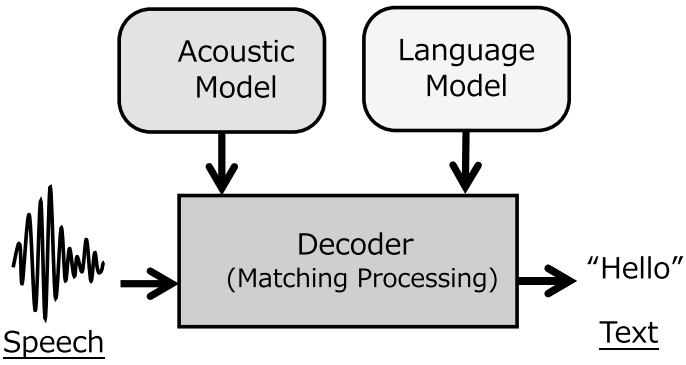

Fig. 1.1 Framework of ASR

1.2.2 Multilingualization of Acoustic Models

1. Acoustic Model

The framework of acoustic models is universal across languages as its primary function is to associate acoustic feature sequences with phoneme sequences. As described in the previous section, the acoustic features are typically short-time physical features of speech that approximate the major elements of sound in which humans identify speech. They include not only static features but also dynamic ones, i.e., the degree of temporal changes. Regardless of language, the absolute values of these features, their fluctuation ranges and rates each falls within a certain range because they are all ruled by the physical and physiological constraints of human speech organs. Therefore, a common language-independent methodology can be applied when extracting and processing acoustic features.

Phonemes are minimum speech units that can be distinguished linguistically, as also mentioned in the previous section, and therefore need to be defined for each language. However, the framework of acoustic modeling can be common. The instantaneous or static property of a phoneme is modeled based on the combinations of short-time acoustic features. On the other hand, modeling along the temporal dimension requires a little ingenuity because phonemes inherently have a high degree of freedom in time. First, the temporal variability among different phonemes: phonemes show a wide variety of intrinsic durations mainly according to the manner of articulation. Sustainable phonemes such as vowels, fricatives, nasals, and trills require a certain amount of time to emit a sufficient amount of acoustic energy that activates the human audition into atmosphere and naturally comprise the longer group, while the shorter group consists of those characterized by rapid motions such as flaps and plosives (excluding pre-plosive closure parts). Second, the temporal elasticity within a phoneme: the duration of the same phoneme can greatly change by various extrinsic factors such as the difference in speakers, speaking rate and style, as well as contextual differences as shown in Table 3.1. The influence on sustainable phonemes are in particular significant. Third, the temporal ambiguity of boundaries of a phoneme: the transition of acoustic features between one phoneme and the adjacent phoneme or the silent part (pause) is not discrete but continuous, because the human speech organ can-not change its position or shape instantaneously. Thus, it is inevitable that there will be an “in-between” section during the transition from phoneme X to Y that cannot be identified as either phonemes from its static features.

For these reasons, as shown in Fig. 1.2, models that allow temporal ambiguity are considered in associating the acoustic feature sequences with phoneme sequences. Specifically, dynamic programming (DP) modeling and hidden Markov model (HMM) are widely used as they can associate the two time series data sets with each other by giving flexibility to the time axis. In particular, HMM is a probabilistic model that allows multiple candidates (e.g., both phonemes X and Y) for the observed acoustic feature, and therefore has a good compatibility with the ambiguous or transitional nature of speech properties mentioned above. This is one

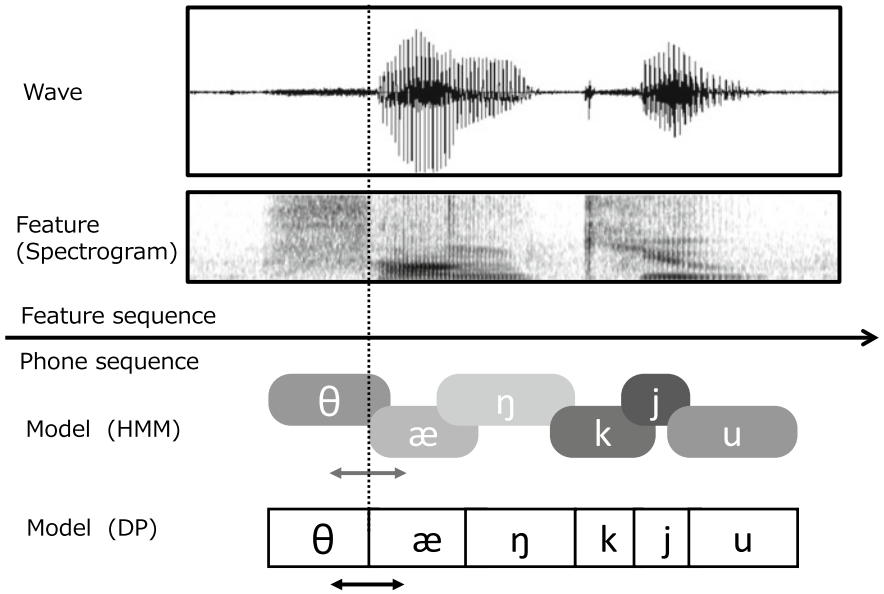

Fig. 1.2 Time sequence matching process of speech feature values and phonemes

reason why HMMs are still being utilized even with the latest advance in machine learning techniques, namely Deep Learning.

However, as a counter effect of ensuring temporal flexibility, the HMM has a disadvantage in the usage of certain kinds of temporal information such as phone durations or rhythms distributed over a relatively wide range of input speech. One of the promising approaches to overcome such disadvantage is end-to-end connectionist temporal classification (CTC) modeling. This allows us to handle global features leading to a higher precision as detailed later in Sect. 2.3.3.

2. Context-Dependent Phones[3]

The use of context-dependent phones enables us to create a highly accurate acoustic model regardless of languages. In the model, a phone is defined differently according to its surrounding phones. The most popular variation is called the triphone model where three consecutive phones are defined as a single unit. For example, the same phone P can be defined in six different ways if it is pronounced immediately after two other phones (a and b) and immediately before three other phones (x, y, and z), they are: a – P + x, a – P + y, a – P + x, b – P + x, b – P + y, and b – P + z (see Fig. 1.3). In theory, a language possessing 50 phones counts a total of 50 × 50 × 50 = 125,000 triphones; however, the actual number is

[3]The term "phone" is occasionally used instead of "phoneme" when any variant of linguistic phonemes, e.g., allophone, is implied.

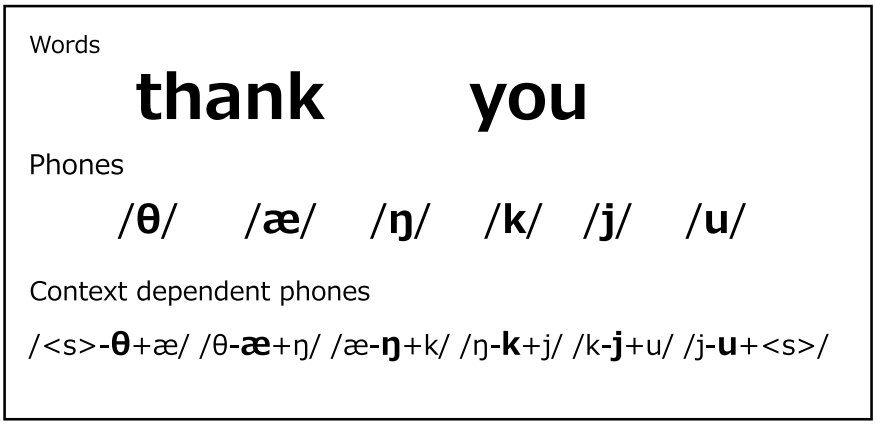

Fig. 1.3 Examples of context-dependent phones. <s> denotes a silent part

usually much smaller than theory dictates—thanks to the phonotactic rules, i.e., language-specific restrictions on what phonemes can exist next to one another.

Context-dependent phones are effective because the acoustic features of a phone are greatly influenced by its preceding and succeeding phones. For example, the movement of the mouth to pronounce a phone varies depending on whether the preceding phone requires the mouth to open wide or not. Any difference in the movement of a speech organ would obviously make a difference in the acoustic features. Therefore, context-dependent phones can provide highly accurate models which cover different varieties of acoustic features due to the context-dependent phonetic coarticulation effects. Note, however, that a large amount of speech data is required to train a large number of phones in this manner. The improvement in ASR accuracy due to the increase of training data is more remarkable with models using deep neural networks (DNNs) than the conventional Gaussian Mixture Model, as the former can handle many-to-many relationships with high precision.

Context-dependent phone modeling not only covers the differences in the order of phones, but also models reductions of articulations as well as unclear utterances. In actual speech, unless done by a professional narrator, it is difficult to clearly utter each and every phone. It is usually the case where one would not have to pronounce each phone clearly to convey his/her message. When exactly will a speech be obscure? It is likely to happen in parts where the intended meaning of an utterance can be linguistically understood without clear articulation and/or in parts that are difficult to articulate. Also, if the listener is native to the language, he/she is accustomed to such unclear or reduced utterances. For example, "lot of" is generally defined as "l ɒ t ə v" in a phone sequence, but the actual speech is likely to be uttered as "l ɒ ɾ ə". As for the reductions of articulations in frequently-used word sequences such as "want to" and "wanna," the latter has become so popular that it is recognized as a different word sequence or a different term by itself. In this way, one does not always pronounce a word with the proper phones which makes it

difficult to associate simple phones with speech. On the other hand, context-dependent phones can be easily matched with actual speech since they are defined depending on the preceding and succeeding phones including their tendencies of becoming unclear. Moreover, since these events are common with all languages, using context-dependent phones would allow speech to be recognized with higher accuracy independent of languages.

It can be said however, that context-dependent phones only consider the neighboring phones and it is difficult to cover the change or reductions of articulations due to a specific word, and especially due to speech features accompanied by large replacements or omissions as seen in spontaneous speech. Therefore, if more detailed phones can be defined in consideration of wider phone alignment, linguistic frequency, and connection between words, a more accurate speech recognition can be realized.

3. Multilingualization

The primary task in the multilingualization of acoustic models is to clearly define the phoneme system of the target language. As mentioned earlier, acoustic models can be respectively modeled according to each phoneme of the target language; however, clear phoneme definitions may not always exist for some languages. Even if a phoneme definition exists, it cannot be verified without knowing the phoneme sequence of the word. Therefore, clarification of the phoneme system is an essential condition in realizing multilingualization of acoustic models, so that pronunciations of all of the words can be defined by phoneme sequences. It is needless to say that some kind of speech corpus (see Sect. 1.1.2) of the target language is necessary to train acoustic models, in addition to an appropriate phoneme system.

1.2.3 Multilingualization of Language Models

A language model is a model for estimating the occurrence probability of a word sequence to a given arbitrary word sequence. The general method is called a statistical language model which performs probability estimation by statistical processing, and the most widely used is the n-gram language model. The n-gram (n = 3 or n = 4 is often used) language model is a model in which the probability $\mathrm{P}(w_1 w_2 \ldots w_n)$ of a word w_n that comes after a word sequence $w_1 w_2 \ldots w_{n-1}$ is calculated, given the conditional probability of the immediately preceding n-1 word, and can be formulated as follows:

$$\mathrm{P}(w_1 w_2 \ldots w_n) = \prod_{i=1}^{n} P(w_i | w_{i-n+1} \ldots w_{i-2} w_{i-1}) \tag{1.1}$$

This probability $\mathrm{P}(w_1 w_2 \ldots w_n)$ is obtained from a large amount of text called (training) corpus, such as collections of example sentences, newspaper articles,

Web pages, and transcriptions of speech data. Therefore, the probability of an n-gram which does not appear in the corpus is 0. In order to avoid this, smoothing is performed to apply a certain probability value to such n-gram. Among some of the smoothing methods such as linear interpolation and maximum entropy method, estimating occurrence probabilities from a lower-order n-grams called back-off smoothing is most widely known. For example, a language model with n = 3, where 3-gram does not appear in the corpus, the occurrence probability of the 3-gram is obtained using that of the lower order (n = 2). In addition, proper nouns such as names of people and companies, cannot achieve high frequency levels from the corpus. In such cases, words can be classified into those with similarities in occurrence tendencies or attributes and a class-based language model—which learns n-grams by class units—can be used. Also, using linear interpolation or the like, a mixture of multiple language models—respectively created from different training corpora with similar topics or writing styles—can be useful. Then, using linear interpolation or the like, the multiple models that have been created are mixed to create one mixture model to be applied for proper nouns. However, these models only estimate the output from the information of history length n, and with languages such as Japanese where a sentence can be very long, and the subject and predicate can be distant from each other, estimation based on topics and utterance styles is challenging. In recent years however, methods using neural networks have been proposed to overcome such challenges and are being put to practical use. For example, recurrent neural network (RNN)—one method of neural networks—hands encoded information of a word history longer than n to the middle layer and feeds back the output to the input of the next intermediate layer enabling outputs to be generated with longer word history taken into account, and has been reported that higher performance can be achieved than the conventional n-gram language models. However, since the smoothing of low-frequency words can be performed more effectively with the n-gram model, neural network-based and n-gram models are sometimes linearly interpolated.

1. Procedural Steps for Creating a Language Model

The language model is created in the procedure depicted in Fig. 1.4.

(a) Preparing a Training Corpus

Prepare a large amount of digitized text (hereinafter referred to as training corpus) written in the recognition target language. The training corpus should preferably be texts from the same task or domain as the target.

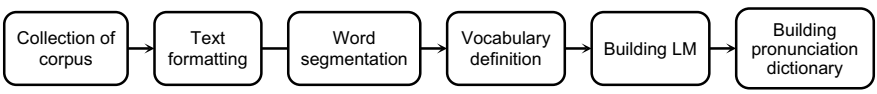

Fig. 1.4 Framework and procedural steps for creating language models

(b) Text Formatting

Delete characters and symbols from the training corpus that are unnecessary for creating the language model. For instance, symbols such as tabs and parentheses which are not uttered in actual conversations should be removed. Arabic numerals should also be spelled out in text to solve the ambiguity on how to read them. Furthermore, when assuming speech input in a sentence-by-sentence manner, the corpus should be divided into sentences.

(c) Word Segmentation

Divide the formatted text into word units. It should be noted however, that in some languages, the pronunciation of numerals (when reading whole numbers) and numeral classifiers may change according to the preceding and/or succeeding words following the sandhi rules. In such case, the word unit is corrected in the post-process of word segmentation. For example, in Japanese, "s a ng b y a k u (三百, or three hundred)" or "i q p o ng (一本, or one, with a numeral classifier which is generally used when counting long objects such as pencils or trees, etc.)" can be divided into "三/百 (three/hundred)" and "一/本 (one/-)," but the pronunciation is different when read together than when read individually, and therefore the word unit must be modified. (Figure 1.5) Any inconsistencies in writing or expressions should also be aligned. For example, variations in letter casing and in abbreviations of the alphabet are recognized as different words in n-gram, as they have the same meaning, are read the same way, but with different spellings. This may lead to a decline in vocabulary coverage and the variance of probability and therefore should be normalized.

(d) Creating a Word List

Create a word list using the word segmentation results. When using large-sized training corpora with large vocabularies, it is common to cut off low-frequency words and determine the vocabulary of the recognition system using only high-frequency words.

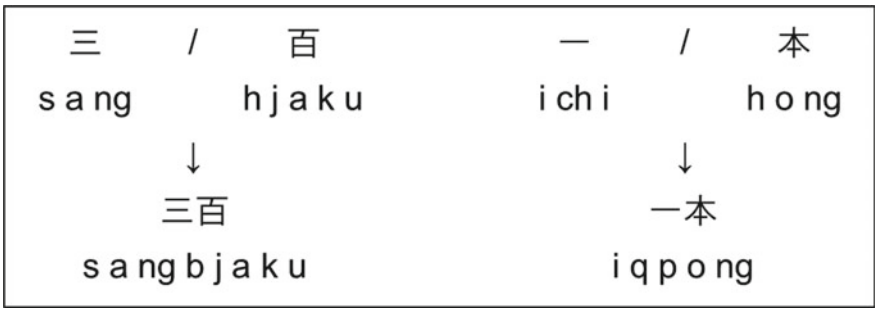

Fig. 1.5 Correction examples of word segmentation results

hello	HH AH0 L OW1
hello	HH EH0 L OW1
thank	TH AE1 NG K
you	Y UW1

Fig. 1.6 Example of a pronunciation dictionary

(e) Creating a Language Model

Create a statistical language model using the above-mentioned high-frequency vocabulary. In recent years, various language modeling tools that facilitate smoothing and model mixing functions such as SRILM (The SRI Language Modeling Toolkit)[4] have been released, and the use of such tools can make language modeling easier.

(f) Creating a Pronunciation Dictionary

Create a dictionary for speech recognition (hereinafter referred to as pronunciation dictionary), which consists of the word list and their pronunciations. In the dictionary, words are typically presented in an orthographic form and pronunciations are presented by phoneme sequences (Fig. 1.6, left and right columns, respectively). A pronunciation dictionary should contain all words to be recognized and all possible pronunciation variations for each of the words, as any "missing" word or pronunciation will be misrecognized. Therefore, it is highly desirable that a pronunciation dictionary is manually created with meticulous care. However, it is not realistic to comprehensively edit the whole dictionary by hand in large-vocabulary systems with several tens or hundreds of thousands of words, which have become common in recent years. A popular solution is to combine automatically-generated grapheme-to-phoneme (G2P) results with the existing manually-created pronunciation dictionary.

2. Multilingualization

The framework and procedures of language modeling are common regardless of languages. Also, preparing a training corpus is language independent. In addition, once the unit used for recognition is determined by word segmentation, word lists and language models can be created—independent of languages—by treating each word as a symbol. On the other hand, text formatting, word segmentation, and

[4] http://www.speech.sri.com/projects/srilm/.

Table 1.2 Numeric expressions by language/region

Country	Value	Date	Currency
United States	1,234,567.89	12-31-01, 12/31/01	$1,234,567.89
Japan	1,234,567.89	2001/12/31	￥1,234,567
United Kingdom	1,234,567.89	31/12/01	£1,234,567.89
Germany	1.234.567,89	2001-12-31, 31.12.01	1.234.567,89€
France	1 234 567,89	31/12/2001	1 234 567,89€
Italy	1.234.567,89	31/12/01	€1.234.567,89
Spain	1.234.567,89	31-12-01	1.234.567,89€

creating pronunciation dictionaries require tuning in accordance with the characteristics of each language.

(a) Text Formatting

Symbols such as tabs and parentheses are to be deleted uniformly regardless of languages. On the other hand, pronunciations of numerals differ by language, and therefore must be tuned individually. For example, how to read numbers may vary among languages, regions, and types of numbers—whether they are expressing dates, money, addresses, phone numbers, whether they should be read digit by digit or as whole numbers, or whether they are cardinal or ordinal numbers. The order of numbers and symbols used together with numbers also vary by language. For example, commas (,) are used to separate digits in English and Japanese, whereas periods (.) are used in German and Spanish, and spaces are inserted in French. In the United States, dates are expressed as "MM-DD-YYYY," with hyphens or slashes, whereas in Japan, it would be "YYYY/MM/DD," and the United Kingdom would use "DD/MM/YYYY," more likely with slashes. Furthermore, the positions of currency symbols—whether to place them in front or after the numbers—also vary among countries (Table 1.2).

It is essential to consider these differences in formats, and different meanings of numbers and symbols in each language and delete or convert them into words. Even when dividing the corpus into sentences, extra care is required for the process because the symbols used for punctuation are different depending on the language. For example, in English, "," are used as delimiters within a sentence and "." are used to separate multiple sentences, but in Japanese, "、" and "。" are used. Such symbols do not exist in the Thai language, which makes it very difficult when dividing them into sentences.

(b) Word Segmentation

Prepare a Word Segmenter for each language. Western languages (e.g., English, French, etc.) whose words are separated by spaces, can easily be divided into word units. Meanwhile, in many Asian languages such as Chinese and Japanese, words are not separated by spaces and statistical methods such as the minimum cost method, HMM, and conditional random field (CRF) are applied to divide the

sentences into words. When the vast majority of high-frequency words are particularly short—comprising only a few phonemes—longer segments such as phrases can be used to prevent misrecognition of non-speech short noises as such short words (insertion errors) or errors in dividing longer words into such short words (segmentation errors). On the other hand, shorter segments such as syllables are considered when n-gram cannot be trained sufficiently from the corpus due to the wide variations in parts of words as observed in conjugational words.

(c) Creating a Pronunciation Dictionary

Since the characters used for each language and their pronunciations are different, it is necessary to create a pronunciation dictionary for each language. The pronunciation dictionary should be created by adding pronunciations to each vocabulary in accordance with the phoneme definition mentioned above. Pronunciations can be automatically generated with G2P by applying ruled-based or statistical methods for languages with less numbers of characters such as English, whereas with languages such as Japanese where there are more and with many different ways to read, it is common to utilize a manually created dictionary.

1.3 Multilingualization of Speech Synthesis

The automatic conversion of text into speech is called "text-to-speech synthesis (TTS)." The term "speech synthesis" is also used in this book interchangeably with "TTS" although "speech synthesis" has a broader meaning to be precise. TTS can be seen as an inverse process of ASR. While it relies on the same types of techniques as ASR in various phases, there are some technical differences between them. Contrasting these two technologies, this section will provide a brief outline of the basic framework of TTS and its multilingualization process.

1.3.1 Basic Framework of Text-to-Speech Synthesis

As shown in Fig. 1.7, a typical TTS system consists of two main processes. The first, which is often referred to as "text analysis," undertakes a linguistic analysis of an input text and outputs an intermediate representation that contains information on how to pronounce the text when it is spoken. A full account of this representation will be given later in Chap. 3. Subsequently, according to the output, the second process generates a speech signal. The generation involves probabilistic "acoustic models" in almost all recent TTS systems based on a statistical method, which the remainder of this section will focus on.

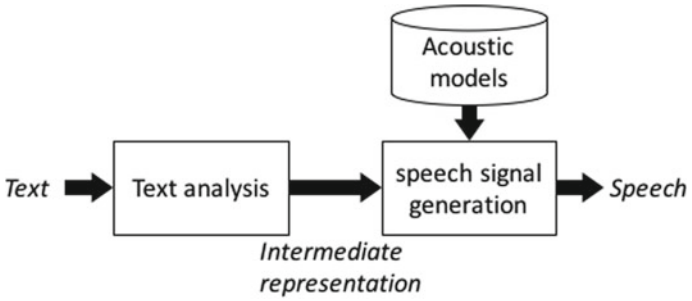

Fig. 1.7 Basic framework of text-to-speech synthesis

1.3.2 Text Analysis

The process of text analysis has some steps in common with that of building language models for ASR (see Sect. 1.2)—text formatting (referred to as "text normalization" in speech synthesis), word segmentation, and grapheme-to-phoneme conversion (relying on a simple dictionary lookup or spelling-based prediction depending on the target language). In addition to these, POS tagging is often incorporated in the analysis process, and for some languages serves as part of the word segmentation process. It should be noted that the whole text analysis process must be executed at runtime in real-time operation for TTS. All the information obtained in the linguistic analysis process is integrated into the intermediate representation, i.e., the output of the text analysis process.

Text analysis needs to be developed depending on the linguistic properties of each individual language. Some of the steps above are clearly language-specific while others can be used commonly among several languages. Text normalization is principally language- or region-specific as in the case of building language models for ASR. Word segmentation (or morphological analysis) is essential in processing languages whose written text has no word delimiters (e.g., Japanese and Chinese). Some of the sophisticated algorithms for the process are universally applicable to such languages. Grapheme-to-phoneme conversion can be quite straightforward for some languages with phonographic writing systems (e.g., Vietnamese and Spanish), whereas the conversion must fully rely on the pronunciation dictionaries for languages with logographic writing systems (e.g., Chinese). Apart from these steps, the intermediate representation includes information that is obviously language-dependent, such as phonemes and lexical accents, and hence needs to be designed specifically for each language (see Chap. 3 for details).

1.3.3 Speech Signal Generation

Upon receiving the intermediate representation, the second process synthesizes speech based on the acoustic models. The basic framework of the acoustic models used in TTS is almost the same as those used in ASR. They are generative models such as HMMs and a DNN, which have been trained in advance with a speech corpus (or training data), independent of the TTS runtime (i.e., offline). A detailed description of the acoustic models used for TTS is beyond the scope of this chapter and can be found in Chap. 3.

The main tasks for the multilingualization process of speech signal generation include: (1) creating a speech corpus of a new language, which further involves (2) recording a certain amount of human speech in that language, and (3) training of the models using the speech corpus. This is clearly because, as in the case of ASR, the speech corpus itself and acoustic models retrieved from the corpus are fully language-dependent while the generation process itself (i.e., the algorithm) is mostly language-independent.

Reference

1. Handbook of the International Phonetic Association: A Guide to the Use of the International Phonetic Alphabet. Cambridge University Press (1999)

Chapter 2
Automatic Speech Recognition

Xugang Lu, Sheng Li, and Masakiyo Fujimoto

Abstract The main task of automatic speech recognition (ASR) is to convert voice signals to text transcriptions. It is one of the most important research fields in natural language processing (NLP). With more than a half century of endeavor, the word error rate (WER), which is a metric unit for transcription performance, has significantly been reduced. Particularly in recent years, due to the increase of computational power, large quantity of collected data, and efficient neural learning algorithms, the dominant power of deep learning technology further enhanced the performance of ASR systems to a practical level. However, there are still many issues that need to be further investigated for these systems to be adapted to a wide range of applications. In this chapter, we will introduce the main stream and pipeline of ASR frameworks, particularly the two dominant frameworks, i.e., Hidden Markov Model (HMM) with Gaussian Mixture model (GMM)-based ASR which dominated the field in the early decades, and deep learning model-based ASR which dominates the techniques used now. In addition, noisy robustness, which is one of the most important challenges for ASR in real applications, will also be introduced.

X. Lu (✉) · S. Li · M. Fujimoto
Advanced Speech Technology Laboratory, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology, Kyoto, Japan
e-mail: xugang.lu@nict.go.jp

S. Li
e-mail: sheng.li@nict.go.jp

M. Fujimoto
e-mail: masakiyo.fujimoto@nict.go.jp

Y. Kidawara et al. (eds.), *Speech-to-Speech Translation*, SpringerBriefs in Computer Science, https://doi.org/10.1007/978-981-15-0595-9_2

2.1 Background of Automatic Speech Recognition

Automatic speech recognition (ASR) is a technology which converts voice into text transcriptions and is one of the core techniques in man-to-machine communications. In recent years, several applications have extensively used ASR-related speech technologies for information access and speech-to-speech translation services. To name a few, Apple's speech assistant Siri, Amazon's home management service Alexa/Echo, Google's smart search and service assistant Google Home, Microsoft's personal assistant Cortana, and NICT's speech translation system VoiceTra.[1] In most of these applications, ASR-related speech communication serves as an efficient and smart interface, and the performance of ASR is essential to the applicability of these services.

The tasks and application scenarios for ASR have changed over the years from simple to complex conditions, for example, from small to large-sized vocabulary, from speaker-dependent to speaker-independent, from read speech to spontaneous speech, and from quiet office rooms to noisy environments. In order to improve the performance of ASR systems in various conditions, techniques have evolved and advanced significantly during the last several decades. For example, simple template matching-based (e.g., dynamic time warping) techniques were developed in the early 1950s to 70s, the popular hidden Markov model (HMM)-based statistic learning techniques were adopted in the 80s to 90s, and the dominant deep learning techniques were invented in recent years. With more than a half-century endeavor, it was only until recently that the ASR technology could see the promising harvests in real applications. The significant improvement in the Word Error Rate (WER) reduction of ASR systems has increased the applicability of the applications. For example, as reported by Microsoft, the WER of 5.1% in ASR achieved with the latest deep learning techniques even outperforms that of human beings (5.9%) on the Switchboard evaluation data corpus [1].

We must admit however, that there is still a long way to go for research and development of ASR to enable speech techniques to be applied in other situations. For instance, in noisy environments, far-field speech, as well as freestyle speech, the ASR performance is still far from applicable while those conditions are likely to be observed in real situations. In addition, how to utilize long context information to improve the ASR accuracy and understand the meaning are some of the challenges that we face. Knowledge from various disciplines, e.g., human speech perception, linguistics, computer sciences, statistics, are necessary to advance the methods and algorithms to deal with the complex application scenarios.

[1] http://voicetra.nict.go.jp/en/.

2.2 Theoretical Framework and Classical Methods

2.2.1 Statistical Framework of ASR

The ASR task can be explained as a mapping task of a spoken audio sequence to a word sequence. Under the statistical framework, the problem can be formulated by maximizing the posterior probability of a word sequence when observing an acoustic sequence:

$$W^* = \arg\max_W P(W|X) \tag{2.1}$$

where $X = [x_1, x_2, \ldots, x_T]$ is a given observation acoustic sequence, $W = [w_1, w_2, \ldots, w_N]$ is a predicted word sequence. With the Bayesian theory, Eq. (2.1) can be further formulated as:

$$W^* = \arg\max_W \frac{P(X|W)P(W)}{P(X)} \propto \arg\max_W P(X|W)P(W) \tag{2.2}$$

where $P(X|W)$ is regarded as the acoustic model, i.e., given a word sequence, the probability of producing the observed acoustic sequence; $P(W)$ is the language model, i.e., the probability of observing the word sequence. The language and acoustic models are trained independently in most conventional algorithms.

Directly optimizing on Eq. (2.2) is a difficult task, but with a divide-and-conquer strategy by taking the hierarchical structure of speech, Eq. (2.2) can be further formulated as:

$$W^* \approx \arg\max_W \sum_L P(X|L)P(L|W)P(W) \tag{2.3}$$

where $P(L|W)$ is the pronunciation model, i.e., given a word sequence W, the probability of generating a pronunciation sequence $L = [l_1, l_2, \ldots, l_M]$ with a set of basic acoustic units (e.g., phonemes). $P(X|L)$ is a generative model, i.e., given the pronunciation acoustic sequence L, the likelihood probability of generating the observed acoustic sequence X. This generative model can be further formulated with a summation of hidden (or latent) state variables as:

$$W^* \approx \arg\max_W \sum_L P(X|S)P(S|L)P(L|W)P(W) \tag{2.4}$$

where $S = [s_1, s_2, \ldots, s_T]$ is the hidden state sequence, and $P(X|S)$ is the likelihood of acoustic observation sequence conditioned on the hidden state sequence.

Most of the conventional algorithms are based on Eq. (2.4). As seen in Eq. (2.4), it involves a complex computation process for sequential probability estimation. In real implementations, the basic formulations are often approximated or simplified

with assumptions or constraints. The most famous solution is the HMM with Gaussian mixture model (GMM)-based approximation (HMM/GMM) framework, i.e., the hidden state sequence probability can be approximated by Markov chain assumptions, while the output probability for each hidden state is approximated using a GMM.

2.2.2 Classical Pipelines for ASR

Under the Bayesian formulation of the HMM/GMM framework as introduced in Sect. 2.2.1, the conventional ASR system is composed of a cascade of several components as illustrated in Fig. 2.1. In this figure, if the components inside the dotted square is modeled and trained as a single model, it will be the end-to-end framework, i.e., directly mapping from acoustic sequence to word sequence (further details will be provided in Sect. 2.3.3).

The components inside the dotted square show the cascade of the hierarchical modeling components of speech recognition which are used in traditional HMM/GMM. The information processing pipelines are as follows:

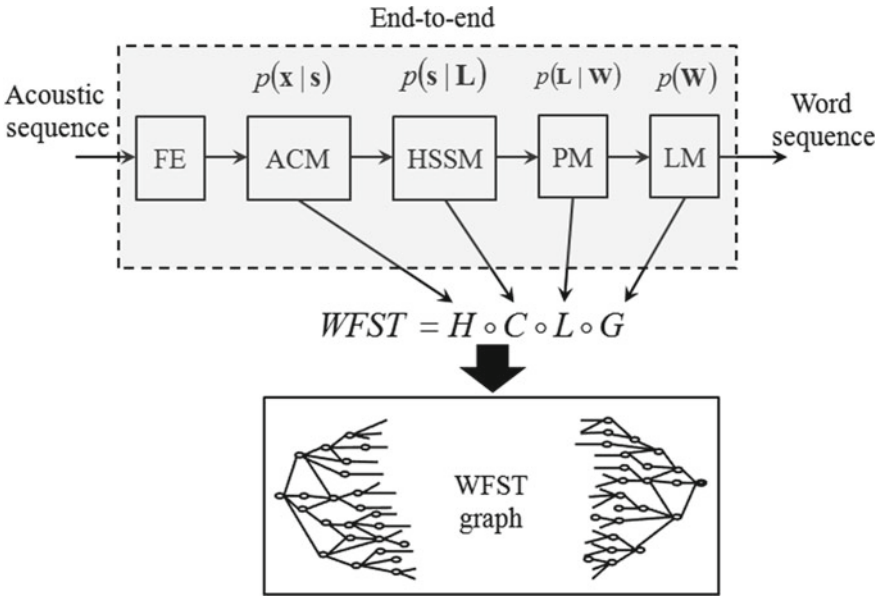

Fig. 2.1 Cascade of pipelines for speech recognition model. FE: Feature extraction, ACM: Acoustic classification model, HSSM: Hidden state sequence model, PM: Pronunciation model, LM: Language model

1. Acoustic speech is encoded as feature vectors with a feature extraction (FE) module which is a deterministic module, e.g., Mel frequency Cepstral Coefficient (MFCC). In end-to-end ASR framework, the processing can be trained.
2. The acoustic classification model (ACM) (e.g., GMM) classifies the feature vector to its hidden state category.
3. The hidden state sequence model (HSSM) is used to model the state transition property of the discrete state labels.
4. The pronunciation model (PM) attempts to model the variations of lexicon pronunciations, and conventionally it is a fixed (deterministic) pronunciation lexicon as used in most ASR systems. In the end-to-end framework, this module also can be trained.
5. The language model (LM) is used to model the word transition property by taking into consideration of the word context history information, e.g., bi-gram and tri-gram models.

In the five modules shown in Fig. 2.1, the ACM (with HSSM), PM, and LM are optimized or trained independently in the conventional HMM/GMM framework. In fact, they can also be jointly optimized. The end-to-end models in particular, are jointly optimized into one framework, i.e., training efficient features with proper pronunciations in utterances for ASR, simultaneously.

In correspondence with the cascade of modules, the recognition (decoding) is also shown in Fig. 2.1. For the convenience of manipulation and extension, the components or pipelines are efficiently described as a weighted finite-state transducer (WFST):

$$HCLG = H \circ C \circ L \circ G \tag{2.5}$$

where G, L, C,H represent the word-level grammar, pronunciation lexicon, context dependency of context dependent phonemes, and the HMM states, respectively; and "$\circ$" is the composition operator. The ASR can be compactly formulated as a composition of several transducers which maps an HMM-state sequence to a word sequence, i.e., finding a solution in a searching space expanded by the decoding graph $HCLG$ [2].

2.3 Deep Learning for ASR

2.3.1 From HMM/GMM to HMM/DNN and Beyond

With the success of deep learning techniques in image processing and recognition, the deep learning algorithms have been applied in ASR and showed dominant effects in improving the performance [3]. Due to the increase in computational power, data corpus, as well as efficient deep learning algorithms, it is safe to say that there has been a paradigm shift from HMM/GMM to HMM/DNN in the ASR field

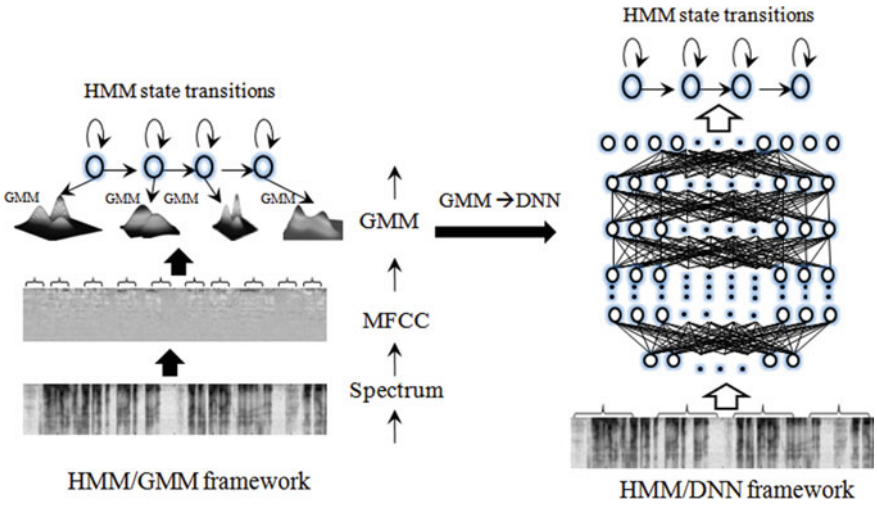

Fig. 2.2 ASR paradigm shift from HMM/GMM to HMM/DNN

in recent years. Figure 2.2 shows this paradigm shift. The left panel shows the conventional framework of HMM/GMM in acoustic modeling. In the HMM/DNN framework, for each state of the HMM, the GMM-based probability approximation is replaced by a DNN-based estimation. In the conventional HMM/GMM, the input to the GMM is a frame-based feature vector, while in the HMM/DNN, the input to the DNN is a feature vector composed of a concatenation of several consecutive frames. In addition, the input to the GMM usually is a dimension-decorrelated vector (e.g., MFCC), while the input to the DNN does not need a decorrelated representation (e.g., Mel frequency filter band spectrum). Aside from the conditional probability estimation for each state, all other processes, e.g., HMM-based state transition model, pronunciation model, and language model are kept the same. Therefore, no big change has been made to the decoding process.

In the HMM, for each hidden state, the probability distribution for observing an acoustic vector can be approximated with parametric models, e.g., GMM. In the GMM, for a given hidden state s_t, the probability of generating an observation acoustic vector x_t can be calculated as:

$$p(x_t|s_t) = \sum_c w_{c,s_t} N\left(x_t; \mu_{c,s_t}, \Sigma_{c,s_t}\right) \tag{2.6}$$

where w_{c,s_t} is the weight of the mixture with constraint as $\sum_c w_{c,s_t} = 1$, c is the number of GMMs, and $N(.)$ is the Gaussian function with mean vector and covariance matrix μ_{c,s_t} and Σ_{c,s_t}, respectively. Equations (2.4) and (2.6) are the fundamental bases of the famous HMM/GMM framework.

In order to use the DNN-based model for likelihood probability estimation in GMM, based on the Bayesian theory, the likelihood probability can be formulated as:

$$p(x_t|s_t) = \frac{p(s_t|x_t)p(x_t)}{p(s_t)} \propto \frac{p(s_t|x_t)}{p(s_t)} \tag{2.7}$$

In the DNN framework, the posterior probability $p(s_t|x_t)$ can be directly estimated based on the feed-forward transformation of DNN. The term $p(s_t)$ is regarded as a scale factor representing the state prior probability. From Eq. (2.7), we can see that the likelihood score can be estimated from the output of a DNN model with a normalization or scale factor [3]. This is the main difference in the frameworks between the HMM/GMM and HMM/DNN.

Although it is assumed that with enough Gaussian kernels, the probability distribution can be accurately approximated by the GMMs, it is difficult for the model to learn a large number of model parameters with good generalization for high-dimensional feature vectors, while the DNN learning framework is good at learning a large number of parameters to approximate the distribution efficiently even with high dimensional feature vectors as inputs. This makes it possible to concatenate several temporal frame windows to utilize the context information in feature representation. In addition, the DNN can approximate an arbitrary probability distribution function with higher accuracy than using GMMs without the GMM assumption of the distribution shape. With these advantages, the DNN can explore nonlinear and complex correlation information in a temporal context window for feature extraction and optimize the feature representation and classification simultaneously.

In DNN modeling, the input window for catching temporal context information is usually fixed, for example, 15 frames of Mel frequency band spectrum or MFCC vectors are concatenated into one long vector to be used as the input. In order to consider a longer or variable temporal window for context information extraction, convolutional neural network (CNN) and recurrent neural network (RNN)-based architectures for acoustic modeling have been proposed. Figure 2.3 shows the basic processing architectures used in FNNs, CNNs, and RNNs. Figure 2.3a is a fully connected feed-forward neural network (FNN), Fig. 2.3b is a convolutional network (CNN) used in deep CNN where the neural weight connections are shared between different convolutional windows, Fig. 2.3c is a recurrent neural network (RNN) where the neural weight connections are shared between different time steps. The CNN is capable of extracting temporal and/or frequency shift invariant features. These invariant features that are related to speaker vocal tract normalization and speaking rate normalization are expected to improve the robustness of ASR. In addition, stacking the CNN layer-by-layer, can extend the width of the input receptive field which catches the hierarchical structure of speech. In CNN-based architecture, the variable temporal context information is obtained at a different level of convolution processing. In order to explore long temporal context information in one processing level for feature extraction and modeling, the RNN architecture is a natural choice where the hidden layer output of the previous process will be used as the input to the current hidden layer (as shown in Fig. 2.3c). However, in model training with error backpropagation (BP) algorithm using

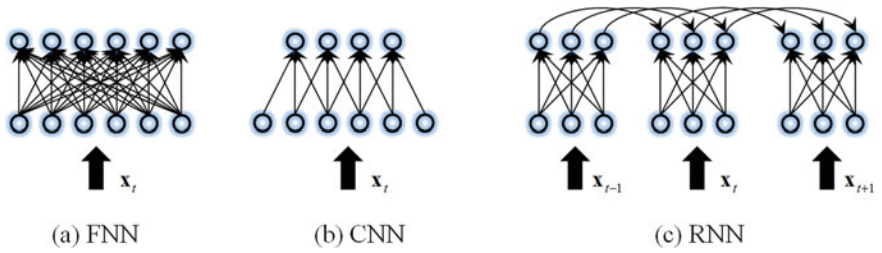

Fig. 2.3 FNN: feed-forward neural network used in DNN (**a**), CNN: convolutional neural network (**b**), RNN: recurrent neural network (**c**)

gradient descent-based optimization, due to the recurrent structure in gradient calculation for long temporal sequence, gradient vanishing or explosion problems always occur which makes it difficult to train the RNN model in its original form. With the long-short term memory (LSTM) [4] and gate recurrent unit (GRU) [5] neural units in the RNN, the RNN can be efficiently trained. Rather than using a single type of neural network structure, combination of different types of neural architectures are proved to improve the performance which takes advantage of the inherent structure of different architectures, for example, the CNN-LSTM-DNN structure which was successfully applied to ASR [6].

2.3.2 From Shallow to Deep

The evolution of the neural network model has gone from shallow (e.g., single-layer perceptron and multi-layer perceptron) to deep (e.g., deep neural network and very deep neural network).

Although the universal approximation theorem [7] of artificial neural networks states that a shallow feed-forward network with a single hidden layer containing a finite number of neurons can approximate any continuous functions, the models widely used at present are mostly deep structured. The reasons are as follows:

1. The deep-stacked nonlinear functions can significantly reduce the requirement of resources. According to the universal approximation theorem, a shallow neural network with the same representation ability requires exponentially many more neurons.
2. GPGPU makes training DNNs with backpropagation algorithm, highly efficient.

Since very deep networks [8] achieved impressive results in image processing and other fields, deeper and deeper structures have been investigated. However, training a very deep network model faces the challenges of gradient vanishing and overfitting problems. For this reason, new architectures, e.g., highway connection [9] and residual connection [10], have been designed to ease gradient-based training of very deep networks. These two types of architectures are shown in Fig. 2.4.

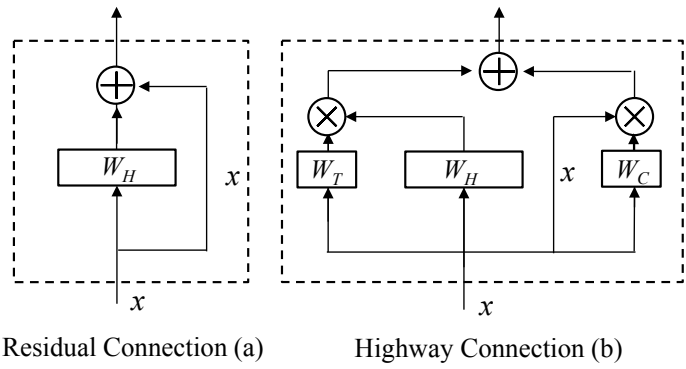

Fig. 2.4 The residual connection (**a**) and highway connection (**b**)

In a normal feed-forward network, input x and output y of a single layer can be expressed as:

$$y = H(x, W_H) \tag{2.8}$$

where H can be a nonlinear transformation in a hidden layer of neural network model (e.g., DNN, CNN, or RNN). When training a model with BP algorithm, the gradient signal will diminish or explode through a very deep stacking of the nonlinearities.

Residual connection structure in Fig. 2.4a is the "$+x$" term as formulated in Eq. (2.9). This structure allows the gradient to pass backwards while skipping all of the intermediate layers and directly reach the bottom layer without being diminished or exploded. With this structure, it is possible to train a stacked convolutional residual network (also known as ResNet) with over 1000 layers.

$$y = H(x, W_H) + x \tag{2.9}$$

Highway connection structure in Fig. 2.4b has similar shortcuts as used in the residual connection and further introduces two learnable gates (transform gate T and carry gate C) to determine which extent each layer should be; a skip connection or a nonlinear connection. The basic calculation unit is formulated as in Eq. (2.10).

$$y = H(x, W_H) \cdot T(x, W_T) + x \cdot C(x, W_c) \tag{2.10}$$

By using these two types of network structures in deep modeling for ASR, promising performances can be obtained.

2.3.3 End-to-End Framework Based on Sequence-to-Sequence Learning

As mentioned in Sect. 2.2.2, in the conventional DNN-based ASR system, each intermediate module is optimized independently. In the end-to-end ASR framework, all intermediate modules can be jointly optimized. The concept of the end-to-end ASR is shown in Fig. 2.5. Figure 2.5a is the conventional DNN-based ASR. Figure 2.5b is the ideal end-to-end ASR system which directly maps an acoustic speech sequence to a word sequence with a unified and jointly optimized model. This does not require the intermediate modules as used in Fig. 2.5a. In current real applications, a language model is still necessary as shown in Fig. 2.5c, whether modeling with sub-word level units (e.g. phonemes or syllables), character level units, or word level units.

There are two most widely-used methods to construct end-to-end ASR systems in the current stage:

1. The Connectionist-Temporal-Classification (CTC) learning [11]

The CTC learning is always used together with bi-directional or uni-directional LSTM (as shown in Fig. 2.6a) and the objective is to automatically learn the end-to-end alignments between speech frames and their label sequence avoiding the segmentation before training.

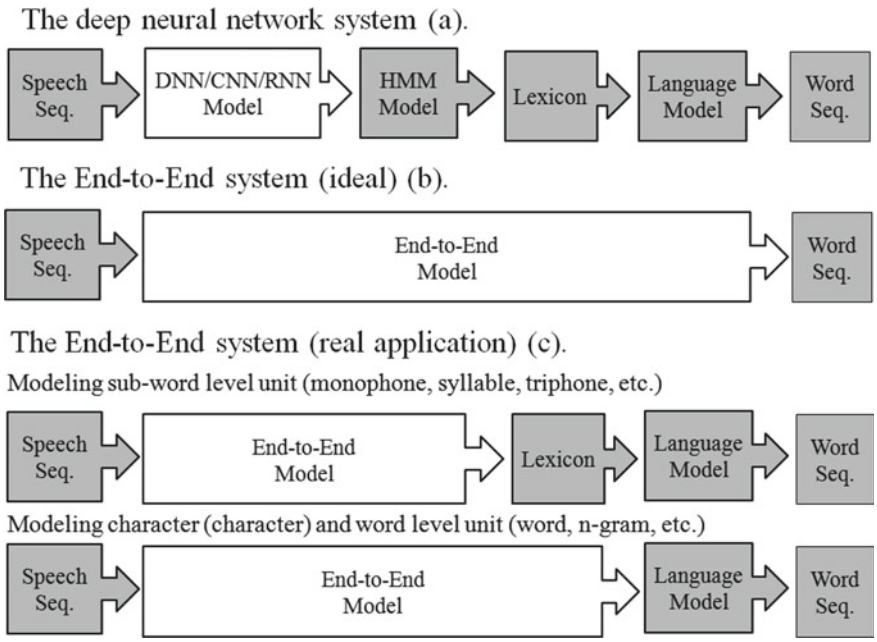

Fig. 2.5 Framework of the conventional DNN system (**a**), the ideal end-to-end system (**b**) and the real applicable end-to-end system in current stage (**c**)

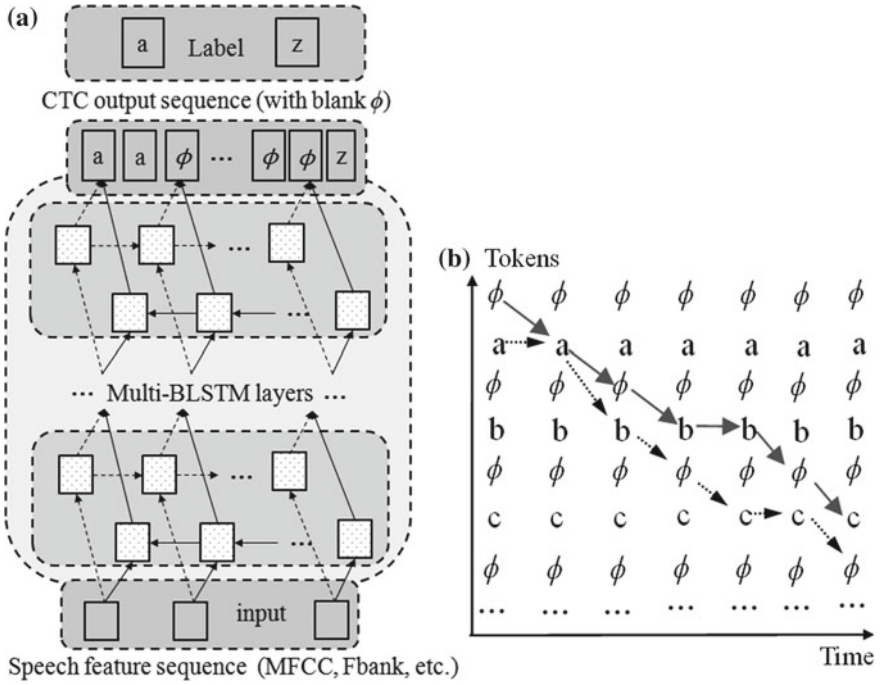

Fig. 2.6 The CTC-based end-to-end model (**a**) and CTC label paths (**b**)

As shown in Fig. 2.6b, "a a b ϕ c c ϕ" and "ϕ a ϕ b b ϕ c" are frame-level token sequences of CTC network outputs. They are mapped into the same label sequence "***a b c***". The blank symbol (ϕ) means no label output. They are inserted between every two labels. The label sequence z is mapped to multiple CTC label paths (denoted as $Path(z)$). A single path of frame-level CTC network output sequence is denoted as p, the likelihood of z can be evaluated as the sum of the probabilities of its corresponding CTC paths:

$$P(z|X) = \sum_{p \in Path(z)} P(p|X) \tag{2.11}$$

where X is the acoustic sequence of a whole utterance, and p is a CTC path corresponding to the label sequence z. By differentiating Eq. (2.11), we can backpropagate training errors and update the network parameters based on a forward-backward procedure [11].

2. The Encoder-Decoder structured neural networks [12]

As the name suggests, the typical Encoder-Decoder structure-based end-to-end system (as shown in Fig. 2.7a) has two components: Encoder and Decoder. The Encoder is a deep (uni-directional or bi-directional) RNN that encodes the speech

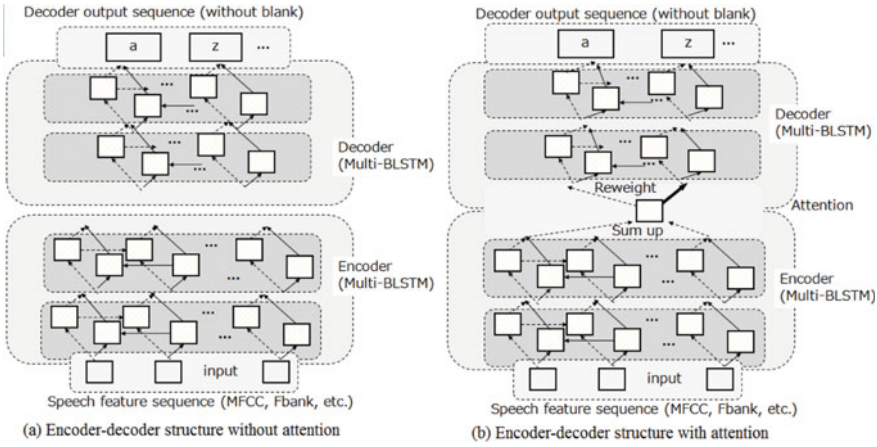

Fig. 2.7 The Encoder-Decoder structured without (**a**) and with attention mechanism (**b**)

signal into a high-level representation, and the Decoder uses a multi-layer RNN to convert the representation into word sequences.

Attention process can be applied to enhance the Encoder-Decoder structure-based end-to-end system [13]. The framework is shown in Fig. 2.7b. In the attention process, the final state of RNN is used as the information summarization of the whole utterance (global representation), and then each frame of the input is reweighted, thus the specific parts of the input can be "focused" on.

Decoding in the end-to-end ASR framework is investigated by many studies. So far, the framework of WFST-based decoding is the most effective method. Based on a constructed WFST graph, a beam search algorithm is applied to find the most probable word sequence. Under the CTC-based framework, in the decoding graph, a token FST T should be composed to the $L \circ G$ FST. The blank symbols and frame-wise token repetitions are removed by the token FST. Then the decoding is performed upon the $T \circ L \circ G$ graph by using the "interpolation-based method" [14] or using the Maximum A Posteriori (MAP) decoding method [15] using the $T \circ S^{-1} \circ L \circ G$ decoding graph which composes a sub-word language model FST S^{-1} with other T, L, G FSTs. Since there are massive blank symbols detected during decoding, the CTC-based decoding can be largely accelerated if the blank frames are skipped and only the key frames are used for decoding.

2.4 Noise-Robust ASR in Real Applications

Although fundamental technologies of state-of-the-art ASR framework have been continuously proposed as described in Sect. 2.3, various crucial problems, e.g., robustness against interference noises and reverberation, still remain as challenges

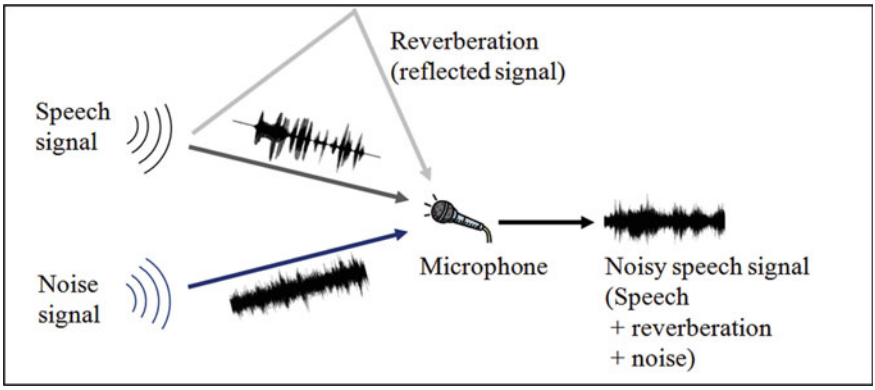

Fig. 2.8 Mixing process of noisy speech signal under the influence of noise and reverberation

in realizing precise ASR in real environments. In real environments, the speech signal uttered by a speaker may be seriously corrupted by interference noises. As shown in Fig. 2.8, when the distance between the speaker and the microphone increases, the speech is easily affected by interference noises with a decrease of the signal-to-noise ratio (SNR) of the recorded speech. In addition, the influence of reflected signals, i.e., that of reverberations become remarkable, and their characteristics are highly fluctuated depending on the conditions of the recording environment, for example, the size of the room, the material of the wall. Due to the noise interference and reverberation as shown in Fig. 2.8, acoustic characteristics between the speech signal uttered by the speaker and the observed signal (noisy speech signal) corrupted by noises and reverberations are considerably mismatched. To ensure the robustness of an ASR against noises and reverberations, it is necessary to adequately reduce this acoustic mismatch.

2.4.1 Techniques of Noise-Robust ASR

There are two types of approaches in realizing a noise-robust ASR. One is front-end processing, including noise reduction and de-reverberation which reduce interference components of noises and reverberations from observed noisy speech signals, and the other is the adaptation of the ACM to the observed environments. Figure 2.9 depicts a classic pipeline architecture of a noise-robust ASR.

In Fig. 2.9, the first module denotes the front-end processing; noise reduction and de-reverberation. In front-end processing, traditional noise reduction techniques, spectral subtraction (SS) [16] and minimum mean-squared error short-time spectral amplitude estimator (MMSE-STSA) [17] have been widely used. Based on the recent deep learning framework, a mapping-based method using a deep de-noising autoencoder (DAE) [18] has been proposed in the research area of

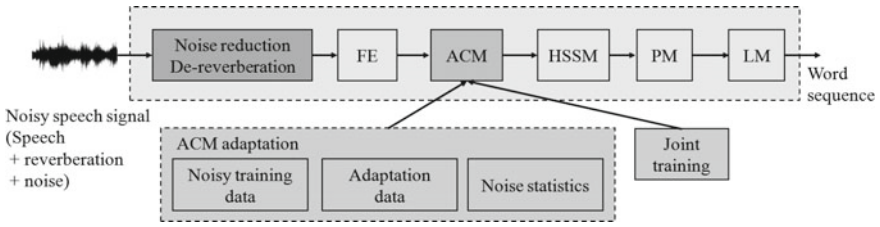

Fig. 2.9 Cascade of pipelines for noise-robust ASR

noise-robust ASR. As a de-reverberation technique, weighted prediction error (WPE) [19] has indicated noticeable ASR improvements under reverberant conditions. While the use of these front-end processing is the simplest way to ensure the noise robustness of an ASR, they sometimes lead to serious performance degradation due to speech distortion caused by estimation errors. This speech distortion is remarkable in single-channel processing. In particular, the recent DNN-based ACM is extremely sensitive to speech distortion. Therefore, to avoid performance degradation of the ASR, it is necessary to carefully choose which techniques to use for noise reduction and de-reverberation. As a solution to this problem, distortionless frameworks have been proposed. Minimum variance distortionless response (MVDR) beamformer [20] that realizes distortionless noise reduction using multiple microphones is widely used in such frameworks, and its effectiveness has been clearly proven by various successful works.

On the other hand, various methods of ACM adaptation have also been proposed. The most representative method of ACM adaptation is multi-condition training. Multi-condition training is a method to learn ACM using training data which consists of noisy speech data. In this method, when the types of noises and SNRs of training data are increased, noise-robust characteristics of the ACM can be improved. However, it is time consuming to train models using data sets combined from various noise types and SNR conditions. Therefore, it is necessary to investigate appropriate types of noises and SNRs in multi-condition training. To cope with unknown noise environments which are not included in training data, techniques of ACM adaptation by using a small amount of environmentally matched data have also been proposed [21]. In addition, noise-aware training [22] that utilizes estimated noise statistics to incorporate noise awareness in ACM learning, has also successfully improved the ASR performance in noise environments. Another recent advanced technique is joint training which combines noise reduction/separation and ACM adaptation with concatenation in a joint training framework [23]. With this framework, it is possible to obtain the ACM which incorporates noise reduction or separation function. Due to the flexibility of the framework, any type of neural network-based noise reduction method, e.g., DNN-based binary masking [24], can be applied.

2.4.2 Evaluation Frameworks of Noise-Robust ASR

Although the representative techniques of noise-robust ASR have been mentioned in Sect. 2.4.1, these methods are often evaluated on different corpus and different evaluation frameworks and therefore, it is difficult to compare the ASR performance between various methods. To simplify the comparison between different methods, some research projects attempt to carefully design noisy speech corpora, ASR tasks, and evaluation criteria as a common evaluation framework.

The SPINE (ASR in noisy environment) project[2] was launched around the year 2000. The corpora of SPINE project were designed as a military communication task including the recognition of spontaneous spoken English dialogue between operators and soldiers.

The AURORA project[3] was also launched in the year 2000. The AURORA project attempted to standardize front-end processing (feature extraction and noise reduction) of ASR service through a mobile phone network, which is referred to as Distributed Speech Recognition (DSR). The AURORA project aimed to develop an advanced ASR front-end that can achieve sufficient ASR performance in real environments. In the AURORA project, four types of corpora, AURORA2, AURORA3, AURORA4, and AURORA5, have been published and distributed through the European Language Resources Association (ELRA). Table 2.1 summarizes the characteristics of each AURORA corpus.

The AURORA project came to an end in 2006. After the AURORA project, a new evaluation project, the Computational Hearing in Multisource Environments (CHiME) challenge[4] has been launched since the year 2011. The CHiME challenge differs from the AURORA project, mainly in (1) using multi-channel speech data, and (2) using real recorded data. It focuses on ASR evaluation in daily environments. Table 2.2 summarizes the characteristics of each CHiME challenge.

The reverberant voice enhancement and recognition benchmark (REVERB) challenge[5] provided LVCSR tasks under various reverberant environments, which includes a simulated task and a real recorded task. In the simulated task, six types of reverberations and a noise were added to the WSJCAM0 corpus, and in the real recorded task, the MC-WSJ-AV corpus, which was recorded under two types of reverberation environments, were used for evaluation. Both the WSJCAM0 and the MC-WSJ-AV corpora have been distributed from the Linguistic Data Consortium (LDC). In both simulated and real recorded tasks, speech data was recorded by using eight microphones. The evaluation tasks were divided into three tasks, 1-channel, 2-channels and 8-channels tasks according to the number of target

[2]SPINE: Speech in noisy environments, http://www.speech.sri.com/projects/spine/.

[3]Aurora speech recognition experimental framework, http://aurora.hsnr.de/index-2.html.

[4]Computational hearing in multisource environments (CHiME) challenge, http://spandh.dcs.shef.ac.uk/projects/chime/.

[5]Reverberant voice enhancement and recognition benchmark (REVERB) challenge, https://reverb2014.dereverberation.com/.

Table 2.1 Summary of the AURORA corpora

	AURORA2	AURORA3	AURORA4	AURORA5
ASR task	Continuous digits	Continuous digits	Continuous read speech	Continuous digits
Language	English	Dutch, Finnish, German, Italian, Spanish	English	English
Vocabulary size	11 digits	11 digits	5000 words	11 digits
Data recording	Simulation	Real recording	Simulation	Simulation
Noise condition	Subway, bubble, car, exhibition, restaurant, street, airport, station	Car noises (Idling, Low-speed, High-speed)	Subway, bubble, car, exhibition, restaurant, street, airport, Station	Car, airport, train street, office, living room
Channel condition or microphone type	2 telephone bands	Close-talking microphone, hands-free microphone	Close-talking microphone, desktop microphone	Close-talking microphone, hands-free microphone
Evaluation track	Matched noise track, mismatched noise track, channel mismatched track	Well-matched track, moderately-matched track, highly-mismatched track	Clean track, noisy track, clean and channel mismatched track, noisy and channel mismatched track	Matched noise and reverberation track, mismatched noise and reverberation track

microphones. The REVERB challenge provided not only ASR evaluation tasks, but also evaluation tasks of processed speech quality by using subjective evaluations based on multiple stimuli with hidden reference and anchor (MUSHRA) test and objective evaluations with various evaluation criteria.

Mutual comparisons using common evaluation frameworks as described in this section brought drastic advances in research on noise-robust ASR. In the AURORA project, many noise reduction techniques and ACM adaptation have been proposed, and research on noise-robust ASR was greatly accelerated. The CHiME challenge and the REVERB challenge were launched at the time of an ASR paradigm shift from HMM/GMM-based systems to HMM/DNN-based systems. In these challenges, researchers attempted to incorporate techniques that have been proposed for the HMM/GMM-based systems in the AURORA project into HMM/DNN-based systems. However, many of them failed and therefore, research on approaches specifically suitable for HMM/DNN-based systems have since been active, such as multi-channel processing, distortionless framework, and multi-condition training,

Table 2.2 Summary of CHiME challenges

	CHiME1	CHiME2	CHiME3	CHiME4
ASR task	Speech command	Speech command, continuous read speech	Continuous read speech	Continuous read speech
Language	English	English	English	English
Vocabulary size	51 grid words	51 grid words, 5000 words	5000 words	5000 words
Data recording	Simulation	Simulation	Simulation, real recording	Simulation, real recording
Noise condition	Living room (multisource noise backgrounds)	Living room (multisource noise backgrounds)	Bus, cafeteria, pedestrian, street	Bus, cafeteria, pedestrian, street
Channel condition or microphone type	Head and torso simulator (binaural recording)	Head and torso simulator (binaural recording)	6 hands-free microphones	6 hands-free microphones
Evaluation track	Clean training (mismatched) track	Small vocabulary mismatched track, large vocabulary mismatched track	1 channel track	1 channel track, 2 channels track, 6 channels track

and have indicated sufficient performance for noise-robust ASR with the HMM/DNN-based systems.

In the future, common evaluation frameworks which are required by real application scenarios will definitely play a large role in accelerating the ASR research to explore new ideas for real applications.

References

1. Xiong, W., Wu, L., Alleva, F., Droppo, J., Huang, X., Stolcke, A.: The Microsoft 2016 conversational speech recognition system. Microsoft Technical Report MSR-TR-2017-39. http://arxiv.org/pdf/1708.06073.pdf
2. Dixon, P.R., Hori, C., Kashioka, H.: Development of the SprinTra WFST speech decoder. NICT Res. J., 15–20 (2012)
3. Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., Kingsbury, B.: Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process. Mag. **29**, 82–97 (2012)
4. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)

5. Cho, K., Merrienboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: encoder-decoder approaches. In: The 8th Workshop on Syntax, Semantics and Structure in Statistical Translation, SSST-8 (2014)
6. Sainath, T.N., Vinyals, O., Senior, A., Sak, H.: Convolutional, long short-term memory, fully connected deep neural networks. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2015). https://doi.org/10.1109/icassp.2015.7178838
7. Csáji, B.C.: Approximation with artificial neural networks. Faculty of Sciences; Eötvös Loránd University, Hungary (2001)
8. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: Proceedings Advances in Neural Information Processing Systems (NIPS) (2012)
9. Srivastava, R.K., Greff, K., Schmidhuber, J.: Training very deep networks. In: Proceedings of NIPS (2015)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016)
11. Graves, A., Fernandez, S., Gomez, F., Shmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the International Conference on Machine Learning (ICML) (2006)
12. Graves, A., Jaitly, N.: Towards end-to-end speech recognition with recurrent neural networks. In: Proceedings of the International Conference on Machine Learning (ICML) (2014)
13. Chorowski, J., Bahdanau, D., Cho, K., Bengio, Y.: End-to-end continuous speech recognition using attention-based recurrent NN: First results. arXiv preprint arXiv:14121602 (2014)
14. Miao, Y., Gowayyed, M., Metze, F.: EESEN: end-to-end speech recognition using deep RNN models and WFST-based decoding. In: Proceedings of IEEE-ASRU (2015)
15. Kanda, N., Lu, X., Kawai, H.: Maximum a posteriori based decoding for CTC acoustic models. In: Proceedings of INTERSPEECH, pp. 1868–1872 (2016)
16. Boll, S.F.: Suppression of acoustic noise in speech using spectral subtraction. IEEE Trans. Audio Speech Signal Process. **27**(2), 113–120 (1979)
17. Ephraim, Y., Malah, D.: Speech enhancement using a minimum mean-square error short-time spectral amplitude estimator. IEEE Trans. Audio Speech Signal Process. **32**, 1109–1121 (1984)
18. Lu, X., Tsao, Y., Matsuda, S, Hori, C.: Speech enhancement based on deep denoising autoencoder. In: Proceedings of Interspeech '13, pp. 436–440, August 2013
19. Yoshioka, T., Nakatani, T.: Generalization of multi-channel linear prediction methods for blind MIMO impulse response shortening. IEEE Trans. Audio Speech Lang. Process. **20**(10), 2707–2720 (2012)
20. Wölfel, M., McDonough, M.: Minimum variance distortionless response spectral estimation. IEEE Signal Process. Mag. **22**(5) (2005)
21. Liao, H.: Speaker adaptation of context dependent deep neural networks. In: Proceedings of ICASSP '13, pp. 7947–7951, May 2013
22. Seltzer, M., Yu, D., Wang, Y.: An investigation of deep neural networks for noise robust speech recognition. In: Proceedings of ICASSP '13, pp. 7398–7402, May 2013
23. Wang, Z., Wang, D.: A joint training framework for robust automatic speech recognition. IEEE/ACM Transa. Audio Speech Lang. Process. (2016)
24. Li, L., Sim, K.C.: Improving robustness of deep neural networks via spectral masking for automatic speech recognition. In: Proceedings of ASRU '13, pp. 279–284, December 2013

Chapter 3
Text-to-Speech Synthesis

Yoshinori Shiga, Jinfu Ni, Kentaro Tachibana, and Takuma Okamoto

Abstract The recent progress of text-to-speech synthesis (TTS) technology has allowed computers to read any written text aloud with voice that is artificial but almost indistinguishable from real human speech. Such improvement in the quality of synthetic speech has expanded the application of the TTS technology. This chapter will explain the mechanism of a state-of-the-art TTS system after a brief introduction to some conventional speech synthesis methods with their advantages and weaknesses. The TTS system consists of two main components: text analysis and speech signal generation, both of which will be detailed in individual sections. The text analysis section will describe what kinds of linguistic features need to be extracted from text, and then present one of the latest studies at NICT from the forefront of TTS research. In this study, linguistic features are automatically extracted from plain text by applying an advanced deep learning technique. The later sections will detail a state-of-the-art speech signal generation using deep neural networks, and then introduce a pioneering study that has lately been conducted at NICT, where leading-edge deep neural networks that directly generate speech waveforms are combined with subband decomposition signal processing to enable rapid generation of human-sounding high-quality speech.

K. Tachibana belonged to NICT at the time of writing.

Y. Shiga (✉) · J. Ni · T. Okamoto
Advanced Speech Technology Laboratory, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology, Kyoto, Japan
e-mail: yoshinori.shiga@nict.go.jp

J. Ni
e-mail: jinfu.ni@nict.go.jp

T. Okamoto
e-mail: okamoto@nict.go.jp

K. Tachibana,
AI System Department, AI Unit, DeNA Co., Ltd., Tokyo, Japan
e-mail: kentaro.tachibana@dena.com

Y. Kidawara et al. (eds.), *Speech-to-Speech Translation*, SpringerBriefs in Computer Science, https://doi.org/10.1007/978-981-15-0595-9_3

3.1 Background

Text-to-speech synthesis (TTS) is a technology of converting written text into speech. In some parts of this book, it is simply referred to as "speech synthesis" without explicitly indicating what the input is. Notice that the broader definition of the term includes some of the technically simpler processes such as recording-reproduction, where pre-recorded voices are simply played back on demand. In contrast to such straightforward processes, TTS requires a much more elaborate mechanism with more advanced techniques to enable computers to read aloud any arbitrary sentence. The recent extensive improvement in synthetic speech quality has brought TTS into wider application areas. A large number of TTS-related products have already appeared on the market, e.g., speech-operated car navigation systems, home assistant devices (so-called "AI speakers") and speech-to-speech translators. The TTS technology is also utilized in almost every ASR-related product presented in Sect. 2.1 for their speech output.

TTS has a long history dating back to the late 1960s. The first practical and commercially-successful TTS system was based on formant speech synthesis [1]. Formant speech synthesis approximates the speech spectrum with several spectral peaks called "formants," which play an important role in human speech perception. The formant-based TTS systems, however, generated speech with many artifacts, which was mainly due to the insufficient accuracy in approximating the detailed speech spectrum. This drawback was overcome later in the late 1980s by concatenative speech synthesis, where synthetic speech was generated by concatenating small fragments of speech. In the early years, concatenative methods adopted speech fragments which were represented by certain acoustic parameters (e.g., the cepstrum) since such parameterization facilitated modifying the fundamental frequency (fo) of a speech. Also, rather small speech units such as diphones were used [2]. However, due to the persistent difficulty in reducing artifacts caused by prosodic modification and spectral interpolation at unit joins, much attention was given to a new type of concatenative synthesis called "unit selection" [3] in the late 1990s. Unit selection speech synthesis is based on a methodology that every synthesis unit of multiple inventory is retrieved selectively from a large-scale speech corpus based on given cost functions, and concatenated with the minimum amount of signal processing. This method produces speech with fewer artifacts when a sufficiently large corpus is available. However, unit selection lacks flexibility in producing various types of timbres or speaking styles since the voice identity of its synthetic speech is greatly dependent upon that of the speech corpus.

Speech synthesis based on hidden Markov models (HMMs) [4], which has been studied since the mid-1990s, has a great deal of flexibility in contrast. In the HMM-based method, the time-varying spectrum and fundamental frequency of speech, as well as the duration of speech segments are modeled simultaneously within the framework of hidden semi-Markov models (HSMMs) [5]. The probabilistic models are trained statistically with a speech corpus in advance and used to synthesize speech waveforms based on the maximum likelihood criterion. Today,

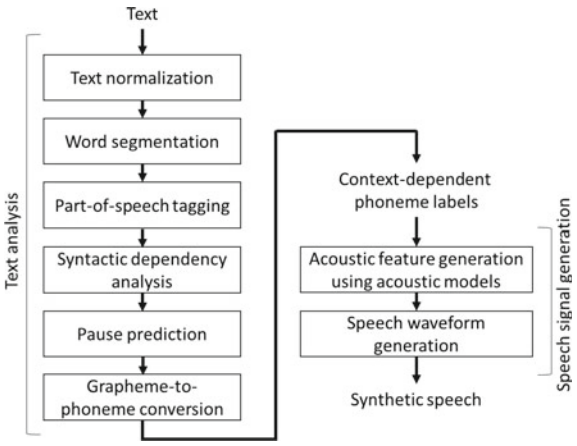

Fig. 3.1 Block diagram of a typical SPSS-based TTS system

such types of speech synthesis based on the statistical training of stochastic generative models are, in general, called "statistical parametric speech synthesis (SPSS)." One of the latest, hottest topics in this field is undoubtedly the introduction of deep learning to the framework of TTS, a detailed explanation of which will appear later in Sects. 3.2 and 3.3.

Figure 3.1 shows a block diagram of a typical SPSS-based TTS system. The system is divided into two main components: text analysis and speech signal generation. As already noted in Sect. 1.3, the text analysis component carries out a linguistic analysis of input text and generates a sequence of phonemes with linguistic context information, which is technically called "context-dependent phoneme labels." In accordance with the sequence of phoneme labels, the speech signal generation component first produces the acoustic features of a target speech and subsequently converts those acoustic features into a speech waveform as an ultimate output of the TTS system.

3.2 Text Analysis for TTS

3.2.1 From Text to an Intermediate Representation

The text analysis component consists of several processes: text normalization, word segmentation, part-of-speech tagging, syntactic dependency analysis, pause prediction, and grapheme-to-phoneme conversion (G2P). As the name implies, the text normalization process normalizes the input text into an appropriate form depending on the context. Typically, numeric expressions (whether "1990" should be transformed as "a thousand nine hundred ninety," "nineteen ninety," or "one nine nine oh"), dates, units, symbols, contracted forms, acronyms, and abbreviations ("Mr.," "Ltd.," and "Co." need to be replaced with "mister," "limited," and "corporation," respectively) are of interest. Specifically, for languages with no word delimiters

Table 3.1 Example of contextual factors that can be used in context-dependent phoneme labels

Type	Description of contextual factors
Lexical	Current phoneme Preceding and following phonemes
	Accent/stress/tone of preceding/current/following syllable
	Part-of-speech of preceding/current/following word
Positional	Position of current phoneme in syllable
	Position of current syllable in word/phrase/utterance
	Position of current word in phrase/utterance Position of current phrase in utterance
	Distance from stressed/accented syllable to current syllable within word/phrase
Length	Number of phonemes in current syllable Number of syllables in current word Number of syllables/words in current phrase
	Number of syllables/words/phrases in utterance
Others	Word/phrase dependency (i.e., modification structure) End tone of phrase

such as Japanese, Chinese, Thai, and Myanmar, a word segmenter (or morphological analysis) is required to identify the boundaries of words within a text. Syntactic dependency is also estimated between adjacent words/phrases. Based on the predicted dependency, the pause prediction process determines where to insert pauses in a sentence. Depending on the language, the G2P process produces information based on the sequence of phonemes with lexical accents/stresses/tones either by referring to pronunciation dictionaries incorporated in the system, or by applying statistical/rule-based letter-to-phoneme conversion or their combination. Further processes are necessary for generating pronunciations for particular languages, e.g., identifying phrase boundaries, determining the pronunciations of heteronyms, processing euphonic changes, devoicing some vowels, etc. Finally, but most importantly, the text analysis component creates and outputs a sequence of phoneme labels with the linguistic context, namely context-dependent phoneme labels. Context is a collection of linguistic information which are likely to affect the acoustic characteristics of a synthetic speech, such as the preceding and following phonemes, the position of syllables within a word, phrase, or utterance, and lexical information such as accents/stresses/tones depending on the target language, which are essential in modeling both the segmental and supra-segmental behaviors of acoustic features. Some of the typically- and practically-used contextual factors are listed in Table 3.1.

3.2.2 Forefront: GSV-Based Method with Deep Learning

Machine learning technology is used to extract linguistic features from input text (hereafter called front-end). In the conventional front-end, a lexicon or dictionary is

necessary for G2P and part-of-speech (POS) tagging, while a language model has to be maintained for word segmentation and syntactic dependency analysis. With the advance in neural network-based machine learning, it has been made possible to model relationships between text and the underlying linguistic features in a sequence-to-sequence manner, i.e., enabling a direct conversion from text to linguistic features.

A neural network-based front-end presented in [6] consists of the following parts: (1) text encoding using global syllable vectors (GSVs) and (2) decoding of the GSVs to obtain linguistic features. GSVs are distributional real-valued representations of syllables and their contextual interactions within normal sentences, which are learned from a large-scale plain text corpus using an unsupervised learning algorithm [7]. The decoding is based on deep bi-directional recurrent neural networks (DBRNNs) which are suitable for solving sequence prediction problems at multiple time scales [8]. GSV-based front-end with DBRNNs can achieve high performance and is fast at running time.

3.2.2.1 Approach Outline

Figure 3.2 shows a schematic diagram of neural network-based text analysis using GSVs as features for the input to DBRNNs. GSVs are learned by GloVe [7] in an unsupervised manner, while DBRNNs are trained with a supervised method.

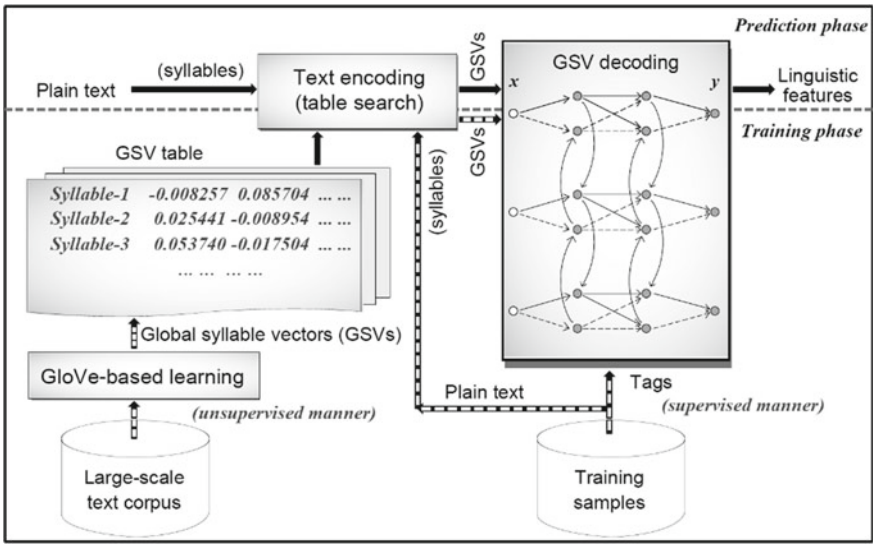

Fig. 3.2 Schematic diagram of GSV-based extraction of linguistic features from text

1. Learning GSVs from a Plain Text Corpus

The statistics of syllable occurrences in a large-scale text corpus is the primary source of information for GloVe to learn space vector representations of syllables. Let X denote the matrix of syllable-syllable co-occurrence frequencies, and x_{ij} occurrence frequency for syllable j occurring in the context of syllable i within a n-syllable symmetric window. GloVe generates d-dimensional syllable vectors by minimizing the following cost function [7].

$$\mathrm{J} = \sum_{i,j=1}^{V} q\left(x_{ij}\right)\left(s_i^T \tilde{s}_j + b_i + \tilde{b}_j - \log x_{ij}\right)^2 \tag{3.1}$$

where V is the size of syllable vocabulary, $s_i \in R^d$ a syllable vector and $\tilde{s}_j \in R^d$ is a separate context syllable vector, b_i and $\tilde{b}_j$ are biases, respectively. $q(x_{ij})$ is a weighting function to avoid frequent co-occurrence from being overweighted [7]. Because X is symmetric, both the syllable vectors s_i and the separate context syllable vectors $\tilde{s}_i$ are equivalent, just with different random initializations. Consequently, GSVs are $s_i + \tilde{s}_i, i = 1, \ldots, V$.

2. DBRNN-Based Prediction

Figure 3.2 shows two-hidden-layer DBRNNs. At time-step t, corresponding to a syllable, each intermediate layer receives one set of parameters from the previous/next time-step in the same layer, and two sets of parameters from the previous bi-directional hidden layers: one input comes from the forward layer and the other from the backward layer. More formally,

$$\vec{h}_t^{(1)} = f\left(\vec{W}^{(1)} x_t + \vec{V}^{(1)} \vec{h}_{t-1}^{(1)} + \vec{b}^{(1)}\right) \tag{3.2}$$

$$\overleftarrow{h}_t^{(1)} = f\left(\overleftarrow{W}^{(1)} x_t + \overleftarrow{V}^{(1)} \overleftarrow{h}_{t+1}^{(1)} + \overleftarrow{b}^{(1)}\right) \tag{3.3}$$

$$\vec{h}_t^{(i)} = f\left(\vec{W}_{\rightarrow}^{(i)} \vec{h}_t^{(i-1)} + \vec{W}_{\leftarrow}^{(i)} \overleftarrow{h}_t^{(i-1)} + \vec{V}^{(i)} \vec{h}_{t-1}^{(i)} + \vec{b}^{(i)}\right) \tag{3.4}$$

$$\overleftarrow{h}_t^{(i)} = f\left(\overleftarrow{W}_{\rightarrow}^{(i)} \vec{h}_t^{(i-1)} + \overleftarrow{W}_{\leftarrow}^{(i)} \overleftarrow{h}_t^{(i-1)} + \overleftarrow{V}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{b}^{(i)}\right) \tag{3.5}$$

where $\mathrm{i} > 1$, arrow $\rightarrow$ indicates the forward direction and $\leftarrow$ the backward direction, h stands for hidden representations, W and V are the weight matrices, b the biases, x_t the input vectors, and $f(x)$ an activation function which is selected according to tasks. The network output, y, at time-step t is

Table 3.2 Experimental setup and results

Task	Training/Dev/Test	Act. function	Output size	Accuracy	CRFs
G2P	44 k/3 k/3.5 k sens	tanh	1390 (pinyins)	**99.05**	98.15
Word segment	95 k/10 k/12 k sens	ReLU	4 (B, I, E, S)[a]	94.31	**95.05**
POS tagging	95 k/10 k/12 k sens	tanh	44 tags	87.33	**90.30**
Phrasing	81 k/11 k/12 k sens	tanh	2 (phrase, other)	**94.61**	93.96
Pause prediction	30 k/3 k/6 k utts	ReLU	2 (pause, other)	**97.34**	97.09

[a]S indicates a mono-syllable token. B, I, and E indicate the syllable positions: at the Beginning, the Inside, and at the End of a token, respectively

$$y_t = g\left(\vec{U}\vec{h}_t^{(L)} + \overleftarrow{U}\overleftarrow{h}_t^{(L)} + c\right) \tag{3.6}$$

where L is the number of layers, U the weight matrices, c the biases, and $g(x)$ the softmax function. Output at time-step t is determined by $argmax(y_t)$.

3.2.2.2 Evaluation

Benchmarking experiments have been conducted using Chinese Treebank 9.0 from the Linguistic Data Consortium (LDC) and in-house speech corpora, which are used to extract pause break samples for supervised learning. Plain text as a sequence of syllables in Chinese is used as the basic unit for prediction. Five main tasks of front-end are evaluated. Table 3.2 tabulates the experimental setup. Training, development, and test sets are disjoint.

1. GSV Learning

A Chinese text corpus amounting to 35 million syllables is used to learn GSVs with varying vector size (25, 50, 100, 200, and 300) and window size (10 and 20 syllables). The size of vocabulary, V, is 5610 (unique syllables).

2. Network Training

The standard multiclass cross-entropy is used as the objective function when training the neural networks. Stochastic gradient descent is used in the experiment with a fixed momentum (0.9) and a small learning rate (0.0001). Weights are updated after mini-batches of 20 sentences. The size of networks changes from 25, 50, 100, to 150 units per hidden layer, but a network has the same number of hidden units across multiple forward and backward hidden layers. The networks are trained with maximum 2,000 epochs, and early stopping is employed: out of all iterations, the model with the best performance for the development set is selected as the final model to be evaluated.

The experimental results are shown at Table 3.2 where the size of GSVs is 50 and the networks have 2 hidden layers and 100 units per layer. Generally speaking, the results are comparable with the baseline conditional random fields (CRFs). For

some tasks like Grapheme-to-phoneme (G2P), the proposed method outperforms the CRFs. Our results also indicate that the effect of GSV size on the performance is slight and that on the window size used for learning GSVs is not significant. Further details can be found in [6].

3.3 Speech Signal Generation

3.3.1 Statistical Parametric Speech Synthesis

Statistical Parametric Speech Synthesis (SPSS) is a framework that generates synthetic speech based on statistical models. One of the advantages that the SPSS has is its high flexibility with respect to voice variation and speech style. On the other hand, overcoming the degradation of speech quality, which is often caused by the differences between the original features and the estimated features by models and the over-smoothing during the modeling process, is an important focus of research. Conventionally, speech synthesis has been actively studied based on the HMM, however in recent years, the center of attention has been replaced by Deep Neural Network (DNN)-based studies [9], as it has been reported by many that DNN outperforms the HMM. Thus, this section will focus on the procedural steps on how to apply DNN in speech synthesis. Figure 3.3 illustrates the overview of the procedure.

3.3.2 Acoustic Model Training

In order to convert text into speech, context-dependent phoneme labels which consist of phonemes and parts of speech are extracted. Then, acoustic features such as spectral envelopes which represent the vocal tracts, and fo which indicates the

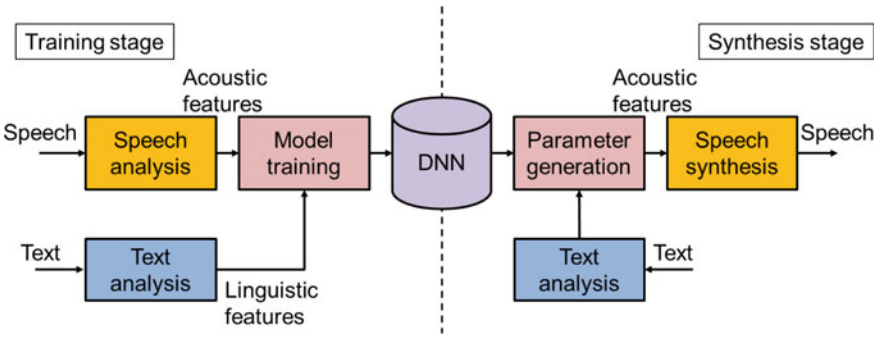

Fig. 3.3 Overview of the DNN-based speech synthesis

pitch of voices, are extracted from the speech. DNN uses the context labels as the input and the acoustic features as the output vectors and trains itself by mapping them out. Specifically speaking, the probability distribution of the acoustic features is modeled instead of directly using the acoustic features themselves as output vectors. While context labels in general are generated in phoneme units, the acoustic features are generated in analysis units called frames. Frame is the minimum unit of the length of a phoneme, and therefore context labels are generated in frame units by indicating the positions of the frames within a phoneme. DNN is trained to minimize the mean squared error between the acoustic feature vectors extracted from the training data and normalized as Z-scores (i.e., mean 0, standard deviation 1), and the vectors predicted by DNN. This is equivalent to modeling the the probability density function (*p.d.f.*) of an original speech parameter vector using the Gaussian distribution as follows:

$$P(\boldsymbol{o}|\boldsymbol{\lambda}) = \prod_{t=1}^{T} \mathcal{N}(\boldsymbol{o}_t|\boldsymbol{\mu}_t, \boldsymbol{U}) \tag{3.7}$$

where $\boldsymbol{o} = \left[\boldsymbol{o}_1^{\mathrm{T}}, \ldots \boldsymbol{o}_t^{\mathrm{T}}, \ldots \boldsymbol{o}_T^{\mathrm{T}}\right]^{\mathrm{T}}$ denotes a speech parameter vector sequence, $\boldsymbol{\lambda}$ denotes DNN parameter set, and $\mathcal{N}(\cdot; \boldsymbol{\mu}_t, \boldsymbol{U})$ denotes a Gaussian distribution with a mean vector $\boldsymbol{\mu}_t$ and a covariance matrix $\boldsymbol{U}$. $\boldsymbol{\mu}_t$ is an unnormalized mean vector given by the speech parameter vector predicted by DNN. $\boldsymbol{U}$ is a global covariance matrix calculated from all frames.

3.3.3 Speech Generation

By using the trained model, the acoustic feature sequences can be predicted from the linguistic feature sequences correspondent to the given text. As mentioned earlier, DNN is capable of predicting the mean vector of the *p.d.f.* of acoustic features for each frame, from the linguistic feature sequences that have been transposed from phoneme units to frames. Then, in consideration of both static and dynamic features, acoustic feature sequences with smoother transitions are generated [10]. Finally, using a "vocoder,[1]" synthetic speech is generated from the predicted acoustic feature sequences.

[1]"Vocoder" is a coined term originated from "voice coder." It generally refers to the whole process of encoding and decoding speech signals for their transmission, compression, encryption, etc., but in the field of TTS, "vocoder" often concerns only the decoding part, where speech waveforms are reconstructed from their parameterized representations.

3.3.4 *Forefront: Subband WaveNet for Rapid and High-Quality Speech Synthesis*

In 2016, deep neural network-based raw audio generative models such as WaveNet [11] and SampleRNN [12] have been proposed, which can synthesize a more natural-sounding speech signals compared with the conventional source-filter model-based vocoders. While vocoders require complicated signal processing for analysis and synthesis with loads of approximations and assumptions, no signal processing—not even Fourier transform—is employed in raw audio generative approaches, which means they can directly model generative probabilities of raw speech waveforms from a speech corpus by using neural networks. As a result, with raw audio generative models, analysis errors and approximation problems which occur in the conventional source-filter model-based vocoders, are not considered an issue. Such models can realize end-to-end speech synthesis from texts to raw speech waveforms within a neural network framework, such as char2wav [13], Tacotron [14], and Deep Voice [15]. In the 2017 European Signal Processing Conference (EUSIPCO), a keynote presentation entitled "Speech synthesis: where did the signal processing go?" was given by Dr. Simon King, in accordance with the latest trend of end-to-end speech synthesis with neural networks. To answer this question, this section introduces the benefits of signal processing for solving two problems in WaveNet.

The conditional probability distribution $p(\boldsymbol{x}|\boldsymbol{h})$ of raw audio waveform $\boldsymbol{x} = [x(1), \ldots, x(T)]$ in WaveNet models, given an additional input h, such as linguistic features for TTS [11] and acoustic features for vocoder [16], can be formulated as:

$$p(\boldsymbol{x}|\boldsymbol{h}) = \prod_{t=1}^{T} p(x(t)|x(1), \ldots, x(t-1), \boldsymbol{h}) \tag{3.8}$$

by a stack of dilated causal convolution layers, which efficiently inputs very long audio samples with just a few layers. In addition, the WaveNet model outputs a categorical distribution instead of a continuous one over to the next sample $x(t)$ with a softmax layer since it is more flexible and can easily model arbitrary distributions, although raw waveform inputs are typically treated as continuous values. In WaveNet, a μ-law encoding, standardized as G.711 [17] is introduced and raw audio waveforms are quantized to 256 possible values. In the training phase, WaveNet can be trained in parallel because all of the $\boldsymbol{x}$ timestamps are known. In the synthesis phase, however, Eq. 3.8 indicates that WaveNet must sequentially synthesize each sample that is fed back to the network in order to synthesize the next one. Therefore, the synthesis speed problem in WaveNet was yet to be solved in 2016, even though parallel computing had been available. The synthesis speed problem is especially severe with higher sampling frequencies for high-quality synthesis, although conventional vocoders can synthesize speech waveforms in real time. Deep Voice introduced smaller networks that quickly predict speech waveforms in real time. However, there is a tradeoff between the synthesis speed and the

synthesized speech quality [15]. In 2017, a parallel WaveNet architecture was proposed and the synthesis speed problem has been solved [18]. However, it still requires a quite complicated neural network training scheme. In addition, Eq. 3.8 indicates that WaveNet models are trained to maximize the signal-to-noise ratio between the speech waveforms of the training set and that of the synthesized ones throughout the frequency range and the prediction errors therefore occur uniformly. As a result, noise components can be heard when their power is greater than the speech components. This problem is also severe with higher sampling frequencies. In parallel WaveNet, additional loss functions were introduced to synthesize high-quality speech waveforms with a sampling frequency of up to 24 kHz.

In order to facilitate rapid synthesis and to realize higher-quality synthesis with sampling frequencies of 32 and 48 kHz, covering the entire human auditory frequency range, a subband WaveNet [19, 20] has been proposed by introducing multirate signal processing [21]. The proposed subband WaveNet architecture is quite simpler than the parallel WaveNet. By introducing a square-root Hann window-based overlapped single-sideband (SSB) filterbank, subband WaveNet can accelerate the synthesis speed and improve the quality of the synthesized speech better than the fullband WaveNet as it improves the prediction accuracy of the WaveNet. A block diagram of the proposed subband WaveNet is described in Fig. 3.4.

In the training stage, fullband speech waveforms $\boldsymbol{x} = [x(1), \ldots, x(T)]$ in the training set are decimated by a factor M and decomposed into N subband streams $\boldsymbol{x}_n = [x_n(1), \ldots, x_n(T/M)]$ with short length T/M and low sampling frequency fs/M by an overlapped SSB analysis filterbank, where N is the number of subbands. Each subband WaveNet network $p_n(\boldsymbol{x}_n|\boldsymbol{h})$ is then separately and efficiently trained by each subband waveform x_n with acoustic features h. In the synthesis phase, each subband stream $\hat{\boldsymbol{x}}_n = [\hat{\boldsymbol{x}}_n(1), \ldots, \hat{\boldsymbol{x}}_n(T/M)]$ is simultaneously generated by the trained network and upsampled by M and each subband waveform with a sampling frequency of fs is obtained by an overlapped SSB synthesis filterbank. Compared with the conventional fullband WaveNet, the proposed subband WaveNet can synthesize N samples at once and realize M times the synthesis speed with parallel computing.

Figure 3.5 shows the spectrograms of an original female test set speech waveform and those estimated by fullband and subband WaveNets for unconditional training and synthesis; where each estimated sample $\hat{\boldsymbol{x}}(t)$ was generated with original past samples $[x(1), \ldots, x(t-1)]$, and the estimated waveform was obtained as $\hat{\boldsymbol{x}} = [\hat{\boldsymbol{x}}(1), \ldots, \hat{\boldsymbol{x}}(T)]$ with a sampling frequency of 32 kHz. The results of the spectral and mel-cepstral distortions indicate that the proposed subband WaveNet can more accurately synthesize speech waveforms than the conventional fullband WaveNet.

The results of a paired comparison listening test between the conventional fullband WaveNet and the proposed subband WaveNet with 21 listening subjects suggested that the proposed subband WaveNet significantly improved the quality of the synthesized speech than the fullband WaveNet [19]. In addition, a subband

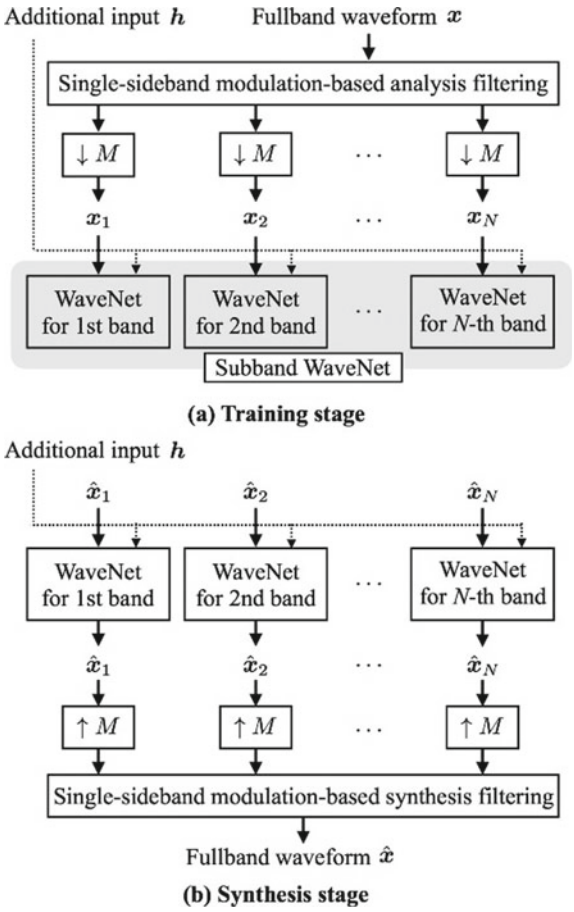

Fig. 3.4 Block diagram of the proposed subband WaveNet

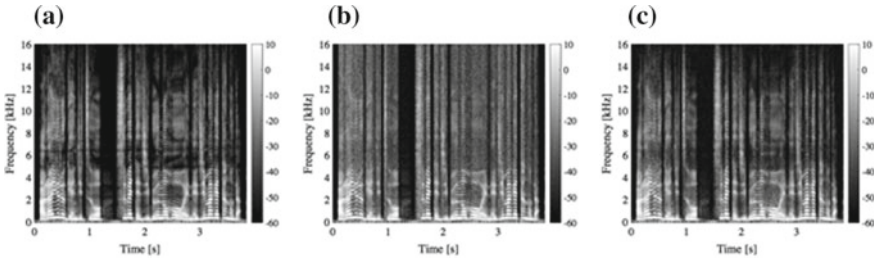

Fig. 3.5 Spectrograms: **a** test set original female speech waveform with a sampling frequency of 32 kHz, **b** estimated by fullband WaveNet, **c** estimated by subband WaveNet

WaveNet vocoder with a sampling frequency of 48 kHz outperformed the conventional source-filter model-based vocoders [20].

Consequently, subband signal processing can accelerate the synthesis speed and improve the quality of the synthesized speech in a raw audio generative model. Future works include the investigations of subband WaveNet architecture-based TTS.

3.4 Future Directions

As we have seen through this chapter, a remarkable progress has been made in the field of both research and development and the rapidly evolving deep learning techniques are expected to resolve many of the existing issues of TTS, in the not too distant future. Some novel approaches that do not consider the aforementioned separation of the TTS process have been proposed very recently, which allow the acoustic features (or speech waveform itself) to be produced directly from text, by fully exploiting the latest deep learning techniques. Such "end-to-end" approaches are still challenging and a detailed description of these is beyond the scope of this book. A list of the references is included at the end of this chapter for those readers who are enthusiastic about exploring such sophisticated approaches in greater depth.

References

1. Klatt, D.H.: Software for a cascade/parallel formant synthesizer. J. Acoust. Soc. Am. **67**(3), 971–995 (1980)
2. Moulines, E., Charpentier, F.: Pitch synchronous waveform processing techniques for text-to-speech synthesis using diphones. Speech Commun. **9**(5), 453–467 (1990)
3. Black, A.W., Campbell, N.: Optimising selection of units from speech databases for concatenative synthesis. In: Proceedings of Eurospeech95, vol. 1, pp. 581–584. Madrid, Spain (1995)
4. Yoshimura, T., Tokuda, K., Masuko, T., Kobayashi, T., Kitamura, T.: Simultaneous modeling of spectrum, pitch and durationin HMM-Based Speech Synthesis. In: Proceeding of Eurospeech, vol. 5, pp. 2347–2350 (1999)
5. Zen, H., Tokuda, K., Masuko, T., Kobayashi, T., Kitamura, T.: A hidden semi-Markov model-based speech synthesis system. IEICE Transactions on Information and Systems, **E90-D**(5), 825–834 (2007)
6. Ni, J., Shiga, Y., Kawai, H.: Global syllable vectors for building TTS front-end with deep learning. In: Proceedings of INTERSPEECH2017, pp. 769–773 (2017)
7. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. http://nlp.stanford.edu/projects/glove/2014
8. Irsory, O., Cardie, C.: Opinion mining with deep recurrent neural networks. In: Proceedings of the 2014 conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 720–728 (2014)

9. Zen, H., Senior, A., Schuster, M.: Statistical parametric speech synthesis using deep neural networks. In: Proceedings of ICASSP, pp. 7962–7966 (2013)
10. Tokuda, K., Yoshimura, T., Masuko, T., Kobayashi, T., Kitamura, T.: Speech parameter generation algorithms for HMM-based speech synthesis. In: Proceedings of ICASSP, pp. 1315–1318 (2000)
11. van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K.: WaveNet: a generative model for raw audio. arXiv preprint arXiv:1609.03499 (2016). (Unreviewed manuscript)
12. Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., Courville, A., Bengio, Y.: SampleRNN: an unconditional end-to-end neural audio generation model. In: Proceedings of ICLR (2017)
13. Sotelo, J., Mehri, S., Kumar, K., Santos, J.F., Kastner, K., Courville, A., Bengio, Y.: Char2wav: end-to-end speech synthesis. In: Proceedings of ICLR (2017)
14. Wang, Y., Skerry-Ryan, R.J., Stanton, D., Wu, Y., Weiss, R., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., Le, Q., Agiomyrgiannakis, Y., Clark, R., Saurous, R.A.: Tacotron: towards end-to-end speech synthesis. In: Proceedings of Interspeech, pp. 4006–4010 (2017)
15. Arik, S.O., Chrzanowski, M., Coates, A., Diamos, G., Gibiansky, A., Kang, Y., Li, X., Miller, J., Ng, A., Raiman, J., Sengupta, S., Shoeybi, M.: Deep voice: real-time neural text-to-speech. In Proceedings of ICML, pp. 195–204 (2017)
16. Tamamori, A., Hayashi, T., Kobayashi, K., Takeda, K., Toda, T.: Speaker-dependent WaveNet vocoder. In: Proceedings of Interspeech, pp. 1118–1122 (2017)
17. ITU-T: Recommendation G. 711. Pulse Code Modulation (PCM) of voice frequencies (1988)
18. van den Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., van den Driessche, G., Lockhart, E., Cobo, L.C., Stimberg, F., Casagrande, N., Grewe, D., Noury, S., Dieleman, S., Elsen, E., Kalchbrenner, N., Zen, H., Graves, A., King, H., Walters, T., Belov, D., Hassabis, D.: Parallel WaveNet: fast high-fidelity speech synthesis. arXiv preprint arXiv:1711.10433 (2017). (Unreviewed manuscript)
19. Okamoto, T., Tachibana, K., Toda, T., Shiga, Y., Kawai, H.: Subband WaveNet with overlapped single-sideband filter- banks. In: Proceedings of ASRU, pp. 698–704 (2017)
20. Okamoto, T., Tachibana, K., Toda, T., Shiga, Y., Kawai, H.: An investigation of subband WaveNet vocoder covering entire auditory frequency range with limited acoustic features. In: Proceedings of ICASSP, pp. 5654–5658 (2018)
21. Crociere, R.E., Rabiner, L.R.: Multirate Digital Signal Processing. Prentice Hall, Englewood Cliffs (1983)

Chapter 4
Language Translation

Kenji Imamura

Abstract This chapter introduces the machine translation part. Machine translation (or text-to-text translation) is one component of speech translation and is used for translating automatic speech recognition (ASR) output into text-to-speech (TTS) input. This section will also explain about neural machine translation, which is the most major translation methodology in recent years.

4.1 Overview of Machine Translation

Machine translation (MT) converts text written in a certain language into that in another language. The input language is called the "source language," and the output language is called the "target language."

Figure 4.1 shows an example of translation from Japanese to English. Lines between the languages indicate the correspondence between words (called "word alignment"). As one may see, machine translation requires a combination of a few different operations:

- If the language is different, words must be converted into different words of the same meaning because the vocabulary is different;
- Words may not always correspond with each other between the source and target languages. Therefore, in some cases, words may need to be eliminated or added, and expressions constructed by multiple words may also have to be converted into different multi-word expressions;
- The sentences of source and target languages must be placed in different word orders.

K. Imamura (✉)
Advanced Translation Technology Laboratory, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology, Kyoto, Japan
e-mail: kenji.imamura@nict.go.jp

Y. Kidawara et al. (eds.), *Speech-to-Speech Translation*, SpringerBriefs in Computer Science, https://doi.org/10.1007/978-981-15-0595-9_4

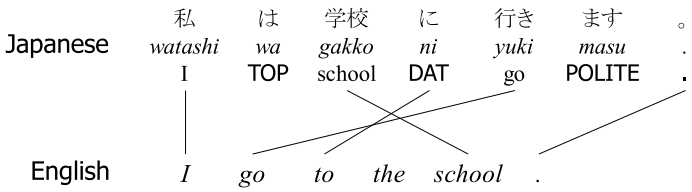

Fig. 4.1 Example of Japanese–English translation

Machine translation performs these operations simultaneously. Most machine translation systems in recent years are based on corpora. Namely, we collect a large number of bilingual sentences as shown in Fig. 4.1 and learn models for translation. We call this set of bilingual sentences "bilingual corpus." The conventional machine translation systems attempted to acquire the above-mentioned operations for translation from corpora. This approach assumes that a sentence translation is constructed from a combination of translated parts. This is called "compositionality." Based on this assumption, statistical machine translation (SMT) [1] was formulated.

However, with the development of deep learning techniques in recent years, methods that are not based on compositionality have been proposed. One typical example is neural machine translation (NMT), which is an end-to-end machine translation method. "End-to-end" means that a source sentence (an input word sequence) can be translated into a target sentence (an output word sequence) with a single model and for this reason, learning and translation processes can be completed in one single step. NMT has become mainstream in speech translation since it has been investigated that higher performance can be achieved compared to SMT.

Many techniques used in neural machine translation are common not only to machine translation but also to neural networks (e.g., the backpropagation). In this chapter, we will focus on techniques that are unique to machine translation, and the explanation of common techniques for neural networks will be left to other books.

In the following sections, we first explain how to handle languages in neural networks (Sect. 4.2). Next, we will talk about the model used for NMT, which is an extension of monolingual language models for two different languages combined (Sect. 4.3). Then, we will review the learning and translation processes in Sect. 4.4.

4.2 Recurrent Neural Network Language Models

When we handle languages with neural networks, there are two points to be noted: (1) the input and output are sequences of variable lengths, and (2) the input and output are symbols represented by words.

4.2.1 Recurrent Neural Networks

The variable length issue can be solved by using a recurrent neural network (RNN). Figure 4.2 shows the simplest structure of an RNN. The input $\boldsymbol{x}$ is converted to output $\boldsymbol{y}$ through a hidden layer $\boldsymbol{h}$. Then, the output from $\boldsymbol{h}$ becomes the input for itself in the next timestep. Along the temporal axis t (the word position in the case of languages), the RNN treats sequences of unlimited length by inputting the output from time $t-1$ to time t. This is represented by the following formulae:

$$h_t = f(Ux_t + Wh_{t-1}), \tag{4.1}$$

$$y_t = g(Vh_t), \tag{4.2}$$

where $\boldsymbol{x}_t$, $\boldsymbol{y}_t$ and $\boldsymbol{h}_t$ denote the states (concretely, vectors of real values) of the input, output, and hidden layers at time t, respectively. $\boldsymbol{V}$, $\boldsymbol{U}$, and $\boldsymbol{W}$ denote the linear weights between the input and hidden layers, the hidden and output layers, and the hidden layer itself, respectively. f and g denote the activation functions of the hidden and output layers, respectively.

When the RNN unit of Fig. 4.2 is expanded according to the temporal axis, the structure becomes what is shown in Fig. 4.3. It is found that the output $\boldsymbol{y}_t$ at time t is influenced by the input $\boldsymbol{x}_1$ of time 1 through multiple hidden layers. However, in a practical situation, it is difficult to learn the network parameters so that $\boldsymbol{x}_1$ affects $\boldsymbol{y}_t$ using simple RNN units. This is called the "vanishing gradient problem," where the network is adapted to the output only using the neighboring inputs because the flexibility of the parameters is very high. Long short-term memory (LSTM) [2] is an RNN unit which contains the input, output, and forget gates, and learns which of the elements should be emphasized or eliminated. By using LSTM as the RNN unit,

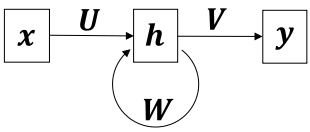

Fig. 4.2 Recurrent neural network

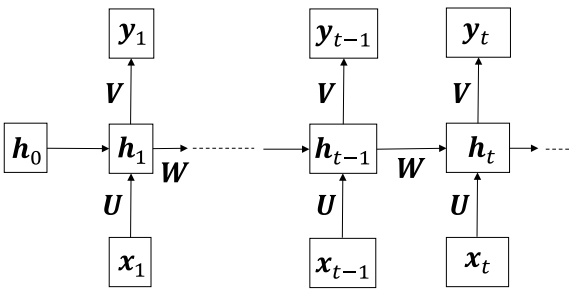

Fig. 4.3 Process of variable-length sequence using recurrent neural network

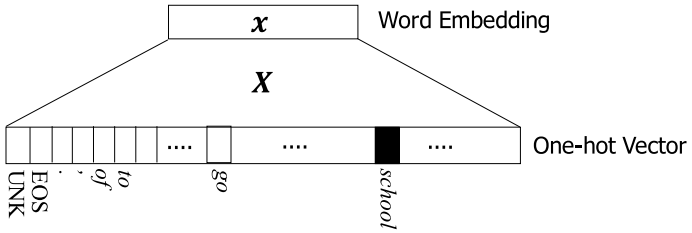

Fig. 4.4 Conversion from one-hot vector to word embedding

the appropriate parameters can be learned even with a long sequence.[1] For further details, please refer to other specialized books on neural networks.

4.2.2 Word Embedding

As for the second issue in handling languages with neural networks—the input and output are symbols—we employ a method so that a symbol is represented by the one-hot vector and converted into a "word embedding (Fig. 4.4)."

The one-hot vector is a high-dimensional sparse vector, whose number of dimensions is equal to the vocabulary size. Each dimension represents a word. The word is represented by setting 1 as the position corresponding to the input word and setting 0 for the others. On the other hand, word embedding is a dense vector (a vector of real values), which is linearly converted from the one-hot vector into a fixed dimension vector. The number of dimensions is usually around 1000 or less. If the word embedding is precisely learned; semantically similar words will come closer to each other in the vector space [3].

When outputting a word, the word embedding is converted into a vector having the same dimensions as the one-hot vector (word distribution). However, it is not one-hot, but a vector that maintains a word generation probability in each dimension. The word distribution is normalized by the SoftMax function to set the sum to 1. Generally, a dimension (i.e., word) having a maximum probability is selected from the word distribution and used as the output.

Note that the vocabulary size is usually limited to tens of thousands due to learning efficiency. When we limit the vocabulary, high frequency words in the training data are selected. This criterion covers many words with a small vocabulary size. Words that are out-of-vocabulary are grouped and associated with a special symbol (often referred to as "UNK") while the network is being learned and applied.

[1]There is another RNN unit called the gated recurrent unit (GRU). Also, there are several versions of LSTM, which are slightly different from each other.

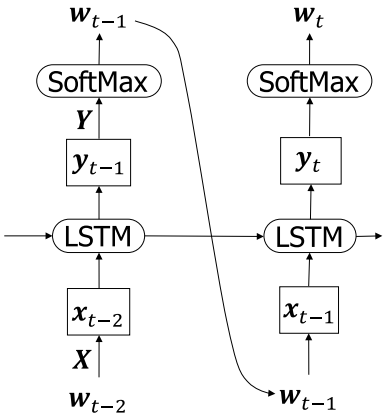

Fig. 4.5 Structure of RNN language model

4.2.3 RNN Language Model

A language model is a model that represents the generation probability of a sentence (a word sequence). The sentence generation probability $\Pr(\boldsymbol{w})$ of a word sequence $\boldsymbol{w}$ is decomposed into the product of the word generation probabilities based on the chain rule.

$$\begin{aligned}\Pr(w) &= \Pr(w_1, w_2, \ldots, w_T) \\ &= \Pr(w_1)\Pr(w_2|w_1)\Pr(w_3|w_2, w_1)\ldots\Pr(w_T|w_{T-1}, \ldots, w_1) \\ &= \prod_{t=1}^{T} \Pr\left(w_t|w_1^{t-1}\right),\end{aligned} \tag{4.3}$$

where T denotes the number of words in a sentence, and w_1^{t-1} denotes the history of the words from the beginning of the sentence to position $t-1$.

The word generation probability depends on its history. N-gram language models, which are used in statistical machine translation, limit the history to the previous $n-1$ words to facilitate the learning.[2] Using RNN, on the contrary, we can implement the language model without any restrictions on history because the RNN itself maintains the whole history in the hidden layers [4]. This model is called RNN language model (RNNLM).

Figure 4.5 illustrates the structure of an RNNLM. An input word is encoded into a word embedding and converted to a word distribution through LSTM units and a SoftMax function. This distribution is regarded as the word generation probability of the language model. A word is selected and output among all words. One characteristic of the RNNLM is that the output word becomes the input word of the next LSTM unit. The generation will be aborted when the end-of-sentence symbol

[2]In statistical machine translation, 3- to 5-gram language models are commonly used.

(EOS) is output. Therefore, the RNNLM contains the EOS symbol in its vocabulary.

The training of the RNNLM is carried out using backpropagation, which is also used for the training of other neural networks. The error, in this case, is the difference between the output word distribution and the one-hot vector of the correct word. During training, the error of the output word distribution $\boldsymbol{w}_t$ is backpropagated to the input, and then the error of the word distribution $\boldsymbol{w}_{t-1}$ is backpropagated. This process is called "backpropagation through time (BPTT)" because the training continues to go back and be applied on the temporal axis. After training the RNNLM, all parameters in the network, i.e., not only the LSTM parameters but also the conversion parameters of the word embeddings are simultaneously learned.

4.3 Models of Neural Machine Translation

4.3.1 Encoder-Decoder Architecture

Machine translation based on neural networks has succeeded in the encoder-decoder architecture [5], which combines two RNNs that handle the source language and the target language, respectively.

A typical structure of the encoder-decoder is shown in Fig. 4.6. The input word sequence is encoded into states (i.e., fixed-length vectors) using the encoder. A "state" is regarded as an extension of the word embedding to a sentence. Using the states as initial inputs, the decoder predicts the output words in order. The translation is completed when the decoder outputs the end-of-sentence symbol. These series of processes can be roughly interpreted that the encoder memorizes the

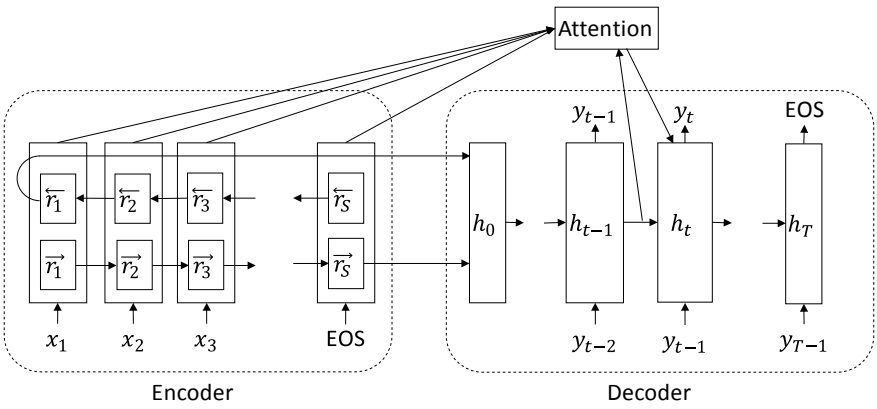

Fig. 4.6 Typical structure of the encoder-decoder (with attention mechanism and bi-directional encoding)

input sentence and the decoder generates the output sentence by playing back the memory.

Since the encoder and decoder are both RNNs, LSTMs are used for their units. It is known that the sentences translated by the encoder-decoder are very fluent because they are essentially the language models.

4.3.2 Attention

Even though LSTM is capable of handling long sequences, the "states" output from the encoder cannot memorize the entire sentence because they are fixed length vectors. The decoder sometimes generates a completely different translation when the input sentence is too long.

To cope with this problem, Bahdanau et al. [6] introduced a mechanism called "attention" to the encoder-decoder architecture. Attention computes the context, which is a weighted sum of the encoder states in each word and uses it as the input to the decoder. By changing the weights, the decoder can focus on a particular word (or words) of the input sentence. The weight of each word is sometimes referred to as "soft alignment" because it is similar to how the word alignment in statistical machine translation becomes probabilistic.

For instance, the global attention proposed by Luong et al. [7] computes an alignment vector $\boldsymbol{a}_t$, whose length is equal to that of the input sentence from the current decoder state $\boldsymbol{h}_t$ and encoder states $\bar{\boldsymbol{r}}$

$$\begin{aligned} a_t(s) &= \mathrm{align}(h_t, \bar{r}_s) \\ &= \frac{\exp(\mathrm{score}(h_t, \bar{r}_s))}{\sum_{s'} \exp(\mathrm{score}(h_t, \bar{r}_{s'}))}, \end{aligned} \tag{4.4}$$

where $\boldsymbol{a}_t$ and $\bar{\boldsymbol{r}}_s$ denote the alignment (weights) and the encoder state at input position s, respectively. Although different score functions have been proposed, the general score can be computed by the following formula:

$$\mathrm{score}(h_t, \bar{r}_s) = h_t^{\mathrm{T}} W_a \bar{r}_s. \tag{4.5}$$

We obtain the weight of each input word by computing the above for all words of the input sentence. The attention mechanism catches a quick glimpse of the input sentence when generating the translation. By introducing the attention mechanism, the system can increase the equivalence of the source and target languages without detracting the fluency of the translation.

4.3.3 Bi-directional Encoding

While the attention mechanism allows the decoder to refer to the context of the input, contexts have directionality; therefore, if the input sentence is encoded simply from the beginning to the end, the context that follows a particular word would not be recognized. To avoid this problem, we introduce another encoder that encodes from the end of the sentence to the beginning, so that the attention can refer to the context from both directions. The attention refers to the input state which is comprised of two concatenated states: the state from the beginning to the end and that from the end to the beginning. This is called "bi-directional encoding."[3] By using the bi-directional encoder, the context of the input sentence can be encoded more precisely. There is also an idea that the decoder should be bi-directional. However, it is not practical because the decoding becomes too complicated although it is known to be effective [8, 9].

4.3.4 Enhancement of Memory Capacity

To improve translation quality, the amount of learning data must be increased and to do so, the capacity of network must be taken into account. In this case, the memory capacity of the encoder and decoder must also be enhanced. There are two approaches to enhance the memory capacity. One is to simply increase the number of dimensions of the encoder and decoder states. Since the learning and translation time increases as the number of dimensions increases, the pros and cons must be considered.

Another approach is to make the encoder and decoder "multi-layered." Since the output of the RNN unit is a word embedding, it can be input to a different RNN unit. By multi-layering the RNN units, more training data can be learned. Indeed, Google Neural Machine Translation (GNMT) [10] constructs the encoder and decoder using 8-layer LSTMs and enabling very large amounts of bilingual texts to be learned.

4.4 Training and Translation

4.4.1 Training

NMT is trained using backpropagation as done in RNN training. The errors are propagated from the last word to the first word of the decoder and furthermore,

[3]When LSTM is used as a RNN unit, it is sometimes called "bi-directional LSTM" or "bi-LSTM."

propagated to the encoder. Because this is an online learning, the network is basically trained sentence by sentence.

NMT is trained by iterating tensor computation. A tensor roughly generalizes a matrix—from a second-order tensor to a higher order. Since tensor operations can be computed quickly using graphics processing units (GPUs), NMT is usually trained on GPUs. In addition, when the GPU power is high, multiple data can be combined into one tensor which can be trained simultaneously in one computation. This multiple datum set is called "minibatch."

The data in a minibatch are bilingual sentences that vary in length. When we construct the minibatch, it is effective to bundle the source and target sentences that have the same length so that the same number of sentences can be processed at every word position.

The appropriate size of the minibatch (how many sentences are processed at a time) depends on the data. If we increase the size, the number of sentences processed per hour is also increased. However, fine tuning becomes difficult because the number of parameter updates in the entire data decreases. Conversely, if we decrease the size, the parameter will be overfit to the last learned data, and the accuracy will degrade.

4.4.2 Beam Search

During the translation process, the decoder outputs a word distribution of each word by computing the network tensors. Then, words with the maximum probabilities are selected from the word distribution. Translation is completed by outputting the selected words in series.

The above process is a 1-best translation using a greedy search algorithm. However, at times, the inferior hypotheses may result to be better than the first hypothesis after all. Therefore, it is better to search for the best sequence throughout the entire time axis. This is called "beam search" and is used with NMT.

The beam maintains the history of output words until the current time and their decoder states. When the beam width is N, the decoder generates N word distributions from each state. Then, it selects the top N words from the word distributions (namely, it generates $N \times N$ hypotheses). A new beam is created by pruning them leaving N hypotheses. The decoding continues onto the next word and stops when all hypotheses in the beam is finished with an end-of-sentence symbol. The translation is the best hypothesis; where the sum of the log-likelihoods—which are logarithms of the word generation probabilities—is the highest.

$$\hat{y} = \mathrm{argmax}_y \sum_t \log \Pr\left(y_t | y_1^{t-1}, x\right), \tag{4.6}$$

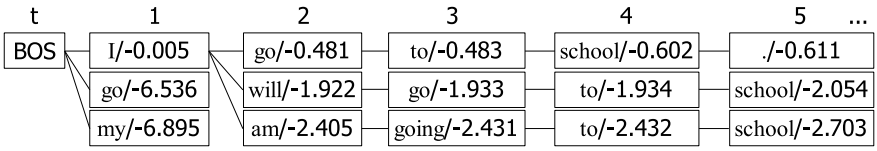

Fig. 4.7 Lattice example of beam search

where $\hat{\boldsymbol{y}}$ denotes the final translation, y_t denotes the output word at the position t, y_1^{t-1} denotes the history of the output words from the beginning of the sentence to the position $t-1$, and $\boldsymbol{x}$ denotes the input word sequence.

Figure 4.7 shows an example of the search lattice for beam search. The input is the same Japanese sentence used in Fig. 4.1, and the beam width is assumed to be 3. Each node at least contains the selected word, its log-likelihood, and its history. While advancing the position t, the beam searcher selects three hypotheses of the highest scores and prunes the others. Finally, "*I go to school.*" is selected in this example.[4]

4.4.3 Ensemble

The accuracy of neural network is known to improve when the outputs from multiple models are averaged [11]. This is called "ensemble." In the case of NMT, an input sentence is independently encoded and decoded using each model, and the output word distributions are averaged. The beam search is applied to the averaged word distribution.

There are different model types for the ensemble; models of another epoch in one training, models of the same structure with the initial parameters being different, and models with different structures (as in the number of layers, their dimensions, and so on). Among them, the most effective combination is the models with different structures. However, the target vocabulary has to be the same because the ensemble averages the output word distributions.

4.4.4 Sub-words

NMT translates a sentence while generating one-hot vectors of the input words and word distributions of the output words. Both of them are vectors whose numbers of dimensions are equivalent with their vocabulary size. Therefore, NMT cannot handle an infinitive vocabulary set, and the size is usually limited to tens of

[4]In contrast to Fig. 4.1, '*the*' was eliminated.

thousands. The out-of-vocabulary words are translated after converting them into "UNK" symbols.

To make the vocabulary size as small as possible, characters should be used as the translation units. However, in general, the characters of the source and target languages do not correspond with each other (except for the languages of the same character set). Therefore, methods that use sub-words have been proposed. The sub-words are units smaller than words and larger than characters.

For instance, Byte Pair Encoding (BPE) [12] first decomposes a word into character symbols, and subsequently create a new symbol by merging two adjacent and highly-frequent symbols. The merging process is aborted when the number of symbol types reaches the desired vocabulary size.

Figure 4.8 is an example where the word '*lower*' is decomposed into sub-words. To clarify the word delimiters in a sub-word sequence, a special character '_' is supplied to the end of the word. In the above example, the word is first decomposed into character symbols. BPE merges two adjacent symbols that are the most frequent in the training corpus and creates a new symbol in series. In this example, four merge operations are applied, and finally the word is segmented into two sub-words '*low*' and '*er_*'.

By introducing sub-words, almost all words can be translated without the "UNK" conversion. Sub-words by BPE do not always become meaningful units. However, this is not a problem in sequence-to-sequence learning based on LSTMs because translation is generated after memorizing the whole sentence, and the attention can refer to multiple symbols at the same time.

4.4.5 Multilingualization

The basic strategy to realize multilingual translation is to collect bilingual corpora of the source and target languages and learn a translator from sentences in the

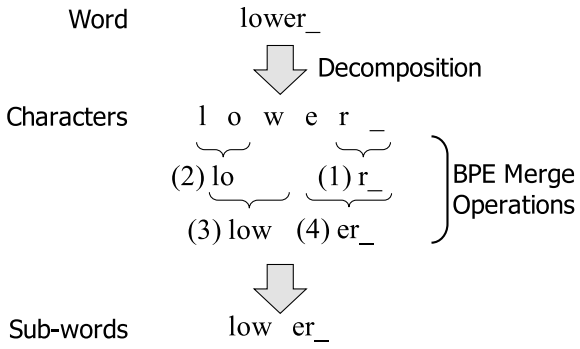

Fig. 4.8 Example of byte pair encoding

corpora (direct translation). However, in some languages, it is impossible to collect bilingual corpora that are large enough to learn the translator. In this case, we define a third language as a pivot—which is different from the source or target language—and translate the sentence using two translators connected in series; from the source to the pivot and from the pivot to the target language. This is called "pivot translation" [13, 14]. The pivot language should be a language in which large corpora exist, i.e., English.

There is another strategy upon using NMT based on the encoder-decoder architecture. That is to learn the encoder and decoder using different language pairs that are not objective. This is called "zero-shot translation" [15].

Figure 4.9 illustrates the differences among direct, pivot, and zero-shot translations. In this figure, zero-shot translation learns a translator from three language pairs (S1–T1, S2–T1, S1–T2) and translates the S2 language into T2. The encoder for the S2 language is learnt from the S2–T1 pair, and the decoder for the T2 language is learnt from the S1–T2 pair. It should be noted that S2 can be translated into T2 although the language pair S2–T2 has not been learnt.

Since zero-shot translation assumes that multiple language pairs are trained at the same time, the models of the translator are required to support multiple languages. The encoder-decoder architecture based on the LSTMs enables the above. For input sentences, the encoder encodes them regardless of the language. However, the decoder must output translations in a specific language. The output language is specified by adding a target language tag to the beginning of the input sentence (Fig. 4.10). Then, the target language information is included in the encoded states, and the decoder can generate a translation according to the tag.

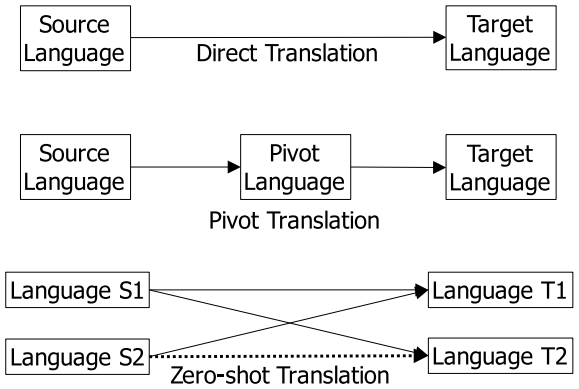

Fig. 4.9 Direct, pivot, and zero-shot translations

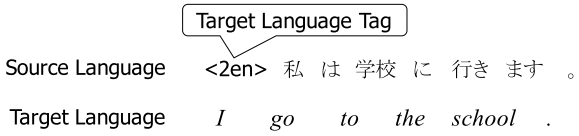

Fig. 4.10 Example of training data of zero-shot translation

Pivot translation can be applied to both SMT and NMT. In addition, the translation quality is better than that of direct translation if the bilingual data is insufficient. On the contrary, zero-shot translation is only applied to NMT, and it can only generate low-quality translation. However, it is drawing attentions as one of the future possibilities of deep learning because it can translate sentences of untrained language pairs.

References

1. Koehn, P.: Statistical Machine Translation. Cambridge University Press (2010)
2. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
3. Mikolov, T., Yih, W.-T., Zweig, G.: Linguistic regularities in continuous space word representations. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2013), Atlanta, Georgia, USA (2013)
4. Mikolov, T., Karafiát, M., Burget, L., Černocký, J., Khudanpur, S.: Recurrent neural network based language model. In: Interspeech '10 Proceedings, Makuhari, Japan (2010)
5. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Proceedings of advances in neural information processing systems 27 (NIPS 2014) (2014)
6. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Proceedings of International Conference on Learning Representations (ICLR 2015) (2015)
7. Luong, T., Pham, H.,D. Manning, C.D.: Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP-2015), Lisbon, Portugal (2015)
8. Liu, L., Utiyama, M., Finch, A., Sumita, E.: Agreement on target-bidirectional neural machine translation. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2016), San Diego, California, USA (2016)
9. Imamura, K., Sumita, E.: Ensemble and reranking: using multiple Models in the NICT-2 neural machine translation system at WAT2017. In: Proceedings of the 4th Workshop on Asian Translation (WAT2017), Taipei, Taiwan (2017)
10. Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Machere, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., Dean, J.: Google's neural machine translation system: bridging the gap between human and machine translation. ArXiv e-prints, 1609.08144 (2016)
11. Hansen, L.K., Salamon, P.: Neural network ensembles. IEEE Trans. Pattern Anal. Mach. Intell. **12**(10), 993–1001 (1990)
12. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Berlin, Germany (2016)
13. Utiyama, M., Isahara, H.: A comparison of pivot methods for phrase-based statistical machine translation. In: Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference, Rochester, New York, USA (2007)

14. Cohn, T., Lapata, M.: Machine translation by triangulation: making effective use of multi-parallel corpora. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL-2007), Prague, Czech Republic (2007)
15. Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., Hughes, M., Dean, J.: Google's multilingual neural machine translation system: enabling zero-shot translation. ArXiv e-prints, 1611.04558 (2016)

Chapter 5
Field Experiment System "VoiceTra"

Yutaka Ashikari and Hisashi Kawai

Abstract This chapter introduces "VoiceTra," a speech-to-speech translation system which utilizes the technologies introduced in the previous chapters. VoiceTra's client application has been released as to the public as a field experiment and has marked a total of 3,921,186 downloads and has collected as many as 178,054,211 speech utterances (as of August 31st, 2019), allowing the system to make continuous improvements. The following sections will cover the system overview, the communications protocols used between the client and server, the user interface of the client application, how the utterances are processed on the speech translation server, and finally the statistical data based on user downloads and logs.

5.1 System Overview

VoiceTra is a network-based multilingual speech-to-speech translation system in which its client application runs on smartphones or tablet devices with either iOS or Android. The client application is connected to the speech translation server via the Internet. The speech translation server is capable of recognizing speech input from users, translating them into other languages, and generating synthesized speech in accordance with the translation results. These high-precision processes generally require a large amount of computing resources and therefore, the speech translation

Y. Ashikari (✉)
System Development Office, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology, Kyoto, Japan
e-mail: ashikari@nict.go.jp

H. Kawai
Advanced Speech Technology Laboratory, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology, Kyoto, Japan
e-mail: hisashi.kawai@nict.go.jp

Y. Kidawara et al. (eds.), *Speech-to-Speech Translation*, SpringerBriefs in Computer Science, https://doi.org/10.1007/978-981-15-0595-9_5

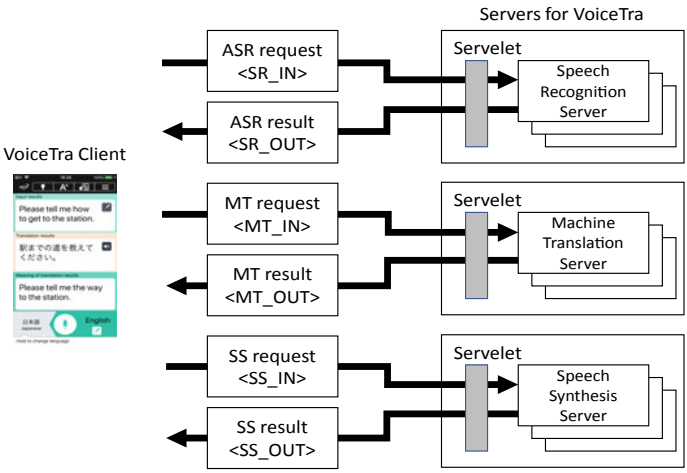

Fig. 5.1 Structure of the VoiceTra system and data communication between the client and server

accuracy of the VoiceTra system is more precise than that of a stand-alone system that performs all processes on a single smartphone.

Figure 5.1 shows the structure of the VoiceTra system. When a user's speech is input to the VoiceTra client, the client sequentially requests the speech translation server to process speech recognition, machine translation, and speech synthesis. The speech translation server then executes these processes and returns the respective results to the client. The client displays the speech recognition and translation results and plays the synthesized speech of the translation result in audio. The speech recognition and synthesis servers are prepared for each language and the translation server is prepared for each language pair. Hence, each server is independently running which makes things easier to update the models and server software. For this reason, adding a new language is also simple. Table 5.1 shows a list of supported languages for VoiceTra (as of August 2019). Speech recognition is available for 18 languages including Japanese, English, Chinese, Korean, Indonesian, Vietnamese, Thai, and Myanmar; and speech synthesis is supported for 16 languages. Machine translation is available between 31 languages which amounts to 465 language pairs in total.

5.2 Communication Protocols for the Client and Server

Speech Translation Markup Language (STML) [1] is used as the communication protocol between the client and server. The server receives the speech input from the client as a speech recognition request <SR_IN>. The recognition result is sent back to the client from the server as <SR_OUT>. The speech recognition result is then sent to the server as a machine translation request <MT_IN>. The translation

Table 5.1 List of supported languages for VoiceTra (as of August 2019)

Languages	Speech input	Speech output	Text input/output
Arabic			Available for all
Brazilian Portuguese	○	○	
Danish			
Dutch			
English	○	○	
Filipino	○	○	
French	○	○	
German	○	○	
Hindi			
Hungarian			
Indonesian	○	○	
Italian			
Japanese	○	○	
Khmer	○	○	
Korean	○	○	
Lao			
Malay			
Chinese (simplified)	○	○	
Mongolian			
Myanmar	○	○	
Nepali	○		
Polish	○		
Portuguese			
Russian	○	○	
Sinhala			
Spanish	○	○	
Chinese (traditional)	○	○	
Thai	○	○	
Turkish			
Urdu			
Vietnamese	○	○	

result is sent back to the client from the server as <MT_OUT>. The translation result is sent to the server as a speech synthesis request <SS_ IN>. The synthesized speech is sent to the client from the server as <SS_OUT>. In addition, the previous translation result is sent to the server as <MT_IN>, translated back into the original language, and sent back to the client as <MT_OUT>. This “back translation” is used to check if the initial translation result contains the intended meaning from the original input.

Each of the requests and processing results are written in XML (Extensible Markup Language). The speech and text data for each language are converted into a MIME (Multipurpose Internet Mail Extensions) format text and transmitted via the Internet using HTTPS (Hyper Text Transfer Protocol Secure).

By using STML, the speech translation server—capable of processing speech recognition, language translation and speech synthesis—can easily be expanded to support new languages or language pairs. Moreover, using STML as the communication protocol allows us to utilize multiple servers developed by different companies and research institutions. Basic tag information included in each STML request and result are as follows:

1. Speech recognition request <SR_IN>

   <MaxNBest>: maximum number of N-best
   <Language>: language code of input speech
   <Audio>: input speech codec
   <SamplingFrequency>: sampling frequency (Hz) of input speech.

   The VoiceTra client records the input speech in 16 kHz 16-bit sampling rate and transmits it to the server after compressing it to 1/4 using ADPCM. Since the maximum number of N-best is normally set to 1, machine translation and speech synthesis are performed in accordance to the best result achieved in speech recognition.

2. Speech recognition result <SR_OUT>

   <NBest>: N-best of the word strings and character strings of speech recognition results

   UTF-8 is used for character encoding.

3. Language translation request <MT_IN>

   <SourceLanguage>: language code of the source language
   <TargetLanguage>: language code of the target language
   <NBest>: N-best of the input word strings and character strings

   The character encoding is UTF-8.

4. Language translation result <MT_OUT>

   <SourceLanguage>: language code of the source language
   <TargetLanguage>: language code of the target language
   <NBest>: N-best of word strings and character strings of the translation result

   VoiceTra only outputs 1-best. The character encoding is UTF-8.

5. Speech synthesis request<SS_IN>

<Language>: language code of the input text
<NBest>: N-best of input word strings and character strings
<Audio>: output speech codec
<SamplingFrequency>: sampling frequency (Hz) of the output speech

In VoiceTra, 1-best is set as the <NBest> of machine translation results. The character encoding is UTF-8.

6. Speech synthesis result <SS_OUT>

<Language>: language code of the output speech
<Audio>: output speech codec
<SamplingFrequency>: sampling frequency (Hz) of the output speech

VoiceTra returns to the client speech in 16 kHz 16-bit sampling rate using ADPCM.

5.3 User Interface

Figure 5.2 is a screenshot of the VoiceTra client application which shows an example of the translation results between Japanese and English. To the left is the main speech translation screen and the language selection screen is shown on the right. The user inputs speech by tapping the microphone button at the bottom center. The application automatically detects the end of the utterance and stops recording. Then, speech recognition, machine translation, and back translation

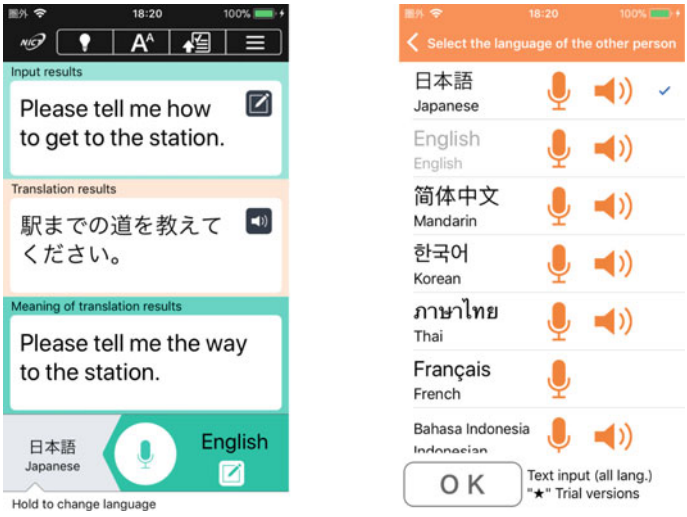

Fig. 5.2 VoiceTra application: main translation screen (left) and language selection screen (right)

Table 5.2 Number of servers used for speech recognition and machine translation

Source language	Speech recognition	Translation target language				
		Japanese	English	Mandarin	Korean	Others (for each pair)
Japanese	40	–	8	8	8	4
English	20	8	–	4	4	4
Mandarin	20	8	4	–	4	4
Korean	20	8	4	4	–	4
Others (for each)	8	4	4	4	4	4

results are displayed in that order from the top and synthesized speech of the translation result is played in audio. For example, in Fig. 5.2, what is displayed at the bottom field is the back translation of the translation result in the middle (in this case Japanese); which has been translated back to the original input language (in this case English). If the back-translation result has the same meaning as the input, the translation is likely to be correct, and by using this, users can check the validity of translation result [2]. The translation direction can be switched by tapping the area where it says "Japanese" at the bottom left corner.

5.4 Speech Translation Server

Speech translation systems are mostly used in face-to-face situations and it is preferred that the speech translation results are simultaneously obtained at the end of each utterance. To realize this, the speech translation process must be performed in real time. The parameters such as beam width have been adjusted for this system so that the mean real-time factor (RTF) for speech recognition becomes 0.5, enabling the results to be obtained as soon as the speaker finishes speaking. The RTF for machine translation is set to 0.05[1] or less. Without taking into account the delay in network, speech translation results can be obtained at a speed of 0.05 times the duration of the utterance, right after the speaker finishes speaking. For example, for an utterance 5 s long, the speech translation result can be obtained within about 0.25 s from the end of utterance.

Table 5.2 shows the number of servers used for speech recognition and machine translation. In the case of performing translation from Japanese to English, the server can process up to 40 simultaneous requests without delay and it can withstand practical use up to an instant maximum of 80 simultaneous requests. This

[1]The value set for when statistical machine translation (SMT) is processed with central processing units (CPUs) only and when neural machine translation (NMT) is processed with CPUs and general-purpose graphics processing units (GPGPUs) combined.

Table 5.3 Number of translation requests between each target/source language from/to Japanese, in percentages

	en	zh	ko	vi	my	th	id	fr	es	other
from ja	62.0	15.8	5.1	3.6	3.0	2.4	1.2	0.9	0.7	5.3
to ja	58.3	18.4	6.5	3.0	4.9	1.8	1.3	0.7	0.4	4.8

Language codes indicate; ja: Japanese, en: English, zh: Chinese, ko: Korean, vi: Vietnamese, my: Myanmar, th: Thai, id: Indonesian, fr: French, and es: Spanish, respectively

Table 5.4 Ratio of input methods (speech, text, and history) used for each language, in percentages

Method	ja	en	zh	ko	vi	my	th	id	fr	es
Speech	69.7	77.4	66.2	74.3	59.6	88.8	63.8	52.9	75.8	54.1
Text	28.1	22.3	33.1	25.4	40.0	10.2	35.9	46.3	23.9	45.3
History	2.2	0.3	0.7	0.3	0.4	1.0	0.3	0.8	0.3	0.6

means that if the average utterance length is 5 s, the server can continue to process about 480 utterances per minute.

5.5 Log Data Statistics

Since its first release on May 11, 2015 up until September 30, 2017 (873 days), the number of downloads of the speech translation app "VoiceTra" marked a total of 1,021,966, of which 57.4% were iOS versions. The total number of translation requests received by the VoiceTra server during the aforementioned period amounted to 35,860,555, of which 52.46% were requests with Japanese being the source language and 31.36% were with Japanese being the target language. Table 5.3 breaks down the ratio into each source/target language and Table 5.4 denotes the number of requests received via different input methods (speech, text, or from history) for each language.

Figure 5.3 shows the 7-day moving average of the number of translation requests received by the VoiceTra server during the same period. The results in the figure reflect the ratio of foreign people that have come to visit or stay in Japan. The number of requests for the Myanmar language exceeded that of Chinese in early June 2017, placing third below Japanese and English, the reason of which is unknown. According to the device location information, it can be predicted that 62% of the input in the Myanmar language has taken place within its own country.

Table 5.5 shows the number of speech recognition requests made for each language on July 5, 2017, a day in which we had a typical number of users, and the average duration as well as the standard deviation values of the speech utterances. Each utterance includes a 0.5 s pause at the beginning and end.

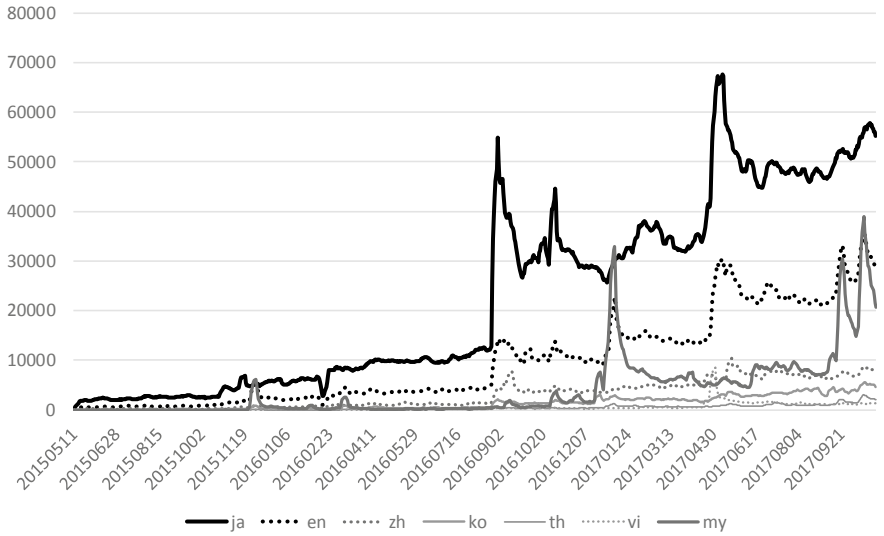

Fig. 5.3 The 7-day moving average of the number of translation requests between May 11, 2015 to September 30, 2017

Table 5.5 Number of speech recognition requests made for each language on July 5, 2017, and the average length and standard deviation values (s.d.) of the utterances in seconds

	ja	en	zh	ko	vi	my	th	id	fr	es
Requests	42,047	22,973	5849	3233	904	11,016	1223	253	696	280
Average length	3.71	3.72	3.42	3.28	3.02	3.34	3.7	3.39	3.58	3.26
s.d.	2.12	2.14	2.18	2.04	1.77	2.02	2.42	1.96	1.86	2.58

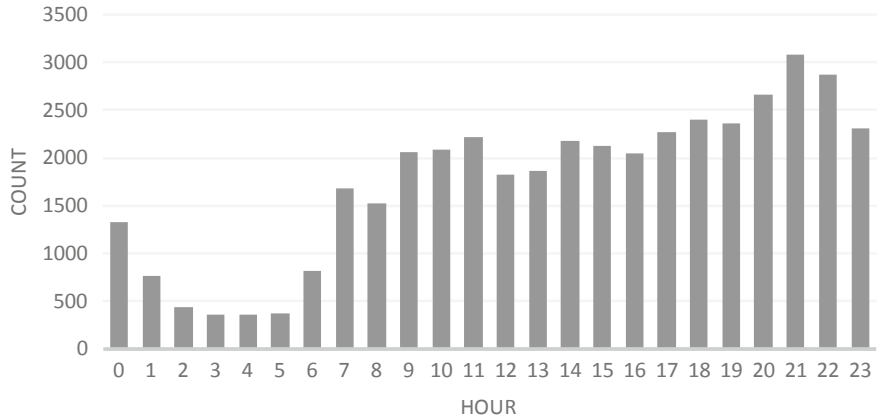

Fig. 5.4 The frequency of Japanese speech recognition requests per hour on July 5, 2017

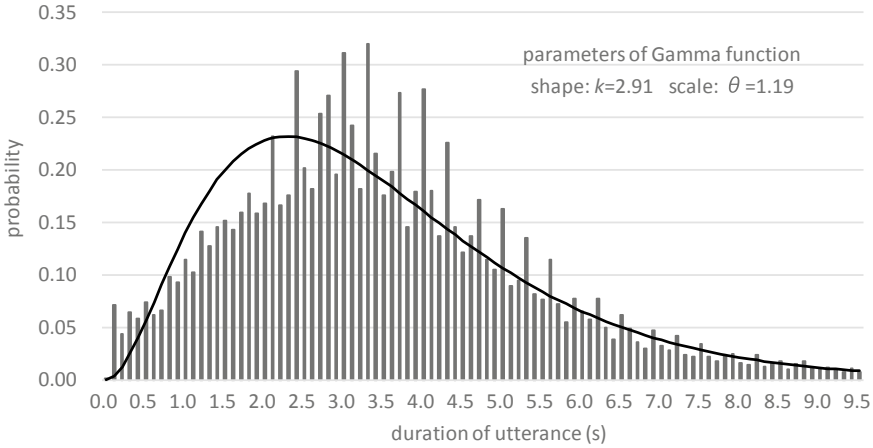

Fig. 5.5 The distribution of Japanese utterance length on July 5, 2017 along with the Gamma distribution fitted to minimize KL-divergence

The frequency of Japanese speech recognition requests made per hour on the same day is shown in Fig. 5.4. While the results show that the frequency was low between 1–6 a.m. and high between 8–9 p.m., no other significant fluctuations were observed.

The distribution of Japanese speech utterance duration for the same day is shown in Fig. 5.5. The curve in the diagram is an approximation to the Gamma distribution.

References

1. Kimura, N.: Speech Translation Markup Language (STML) Version 1.00. Tech. Rep., NICT, Kyoto, Japan (2008)
2. Yasuda, K., Sumita, E., Yamamoto, S., Kikui, G., Yanagida, M.: Real-time evaluation architecture for MT using multiple backward translations. In: Proceedings of RANLP2003, pp. 518–522 (2003)

Chapter 6
Measuring the Capability of a Speech Translation System

Fumiaki Sugaya and Keiji Yasuda

Abstract This chapter introduces an evaluation method for speech translation systems. Since the method estimates the system's capability in a commonly known English proficiency measurement score such as TOEIC, the evaluation results are very comprehensible not only for researchers but also for system users. This chapter also formulates the evaluation errors and discusses the costs necessary for evaluation. Lastly, we provide a solution to the evaluation cost issue by applying an automatic evaluation method.

6.1 Introduction

After many decades of research and development, overseas travelers worldwide now freely access to speech translation systems by smart devices. The speech translation systems are widening the application range, i.e., to questionnaires in the initial medical examination, emergency announcements, shopping, and transportation Q&As. The ultimate question remains: how practical or useful are these systems in real applications?

Many researchers and developers usually evaluate the translation systems with BLEU [1] objective scores and subjective rank evaluations. However, the results are only used to help improve the parameters or algorithms of the translation systems and we would need a new evaluation method to answer the ultimate question. When we hire new employees with foreign language skills for example, we give the applicants a test and often call them in for additional interviews. This

F. Sugaya and K. Yasuda belonged to NICT at the time of writing.

F. Sugaya (✉)
MINDWORD, Inc., Tokyo, Japan
e-mail: fsugaya@mindword.jp

K. Yasuda
Data Science Center, Nara Institute of Science and Technology, Nara, Japan
e-mail: ke-yasuda@dsc.naist.jp

Y. Kidawara et al. (eds.), *Speech-to-Speech Translation*, SpringerBriefs in Computer Science, https://doi.org/10.1007/978-981-15-0595-9_6

idea gives us a hint to relate the capability of the translation system to the test scores for humans.

As for English tests, the Educational Testing Service (ETS) has been studying and providing the Test of English for International Communications (TOEIC)[1] for more than 35 years, which has long been the global standard for measuring English communication skills necessary for employment.

In the following sections, we will propose a translation-paired comparison method which can relate the system's capability to the TOEIC score. In this method, bilingual speakers first compare the system's translation results with that of the examinees with various TOEIC scores. The TOEIC score of the system is then calculated from the comparison data. This method requires two labor costs: the data collection cost of the examinees' translations and the subjective comparison cost by the bilingual speakers. In Sect. 6.2, the proposed method is further described in detail followed by Sect. 6.3 which shows the evaluation results of the actual system. The error analysis is made in Sect. 6.4 and in Sect. 6.5, the evaluation costs are explained. In Sect. 6.6, the automated method of the translation-paired comparison method is proposed. Finally, Sect. 6.7 concludes the sections and describes some of the future tasks.

6.2 Translation-Paired Comparison Method

The translation-paired comparison method [2] is a precise evaluation method for measuring the capability of a speech translation system. This section will explain the method in detail.

6.2.1 Methodology of the Translation-Paired Comparison Method

Figure 6.1 shows a diagram of the translation-paired comparison method in the case of Japanese to English speech translation including speech recognition (SR) and machine translation (MT). The two translation results; one done by humans and the other by the system, are compared and evaluated by bilingual speakers of both languages sentence by sentence, and a winning rate is calculated for all utterances.

First, native Japanese examinees—who are English learners with their capabilities evaluated in TOEIC scores—are asked to listen to a set of Japanese utterances and write the English translation on paper. The examinees are requested to present an official TOEIC score certificate showing that they have taken the test within the past six months. The Japanese is presented twice within one minute with a pause in

[1] https://www.ets.org/about.

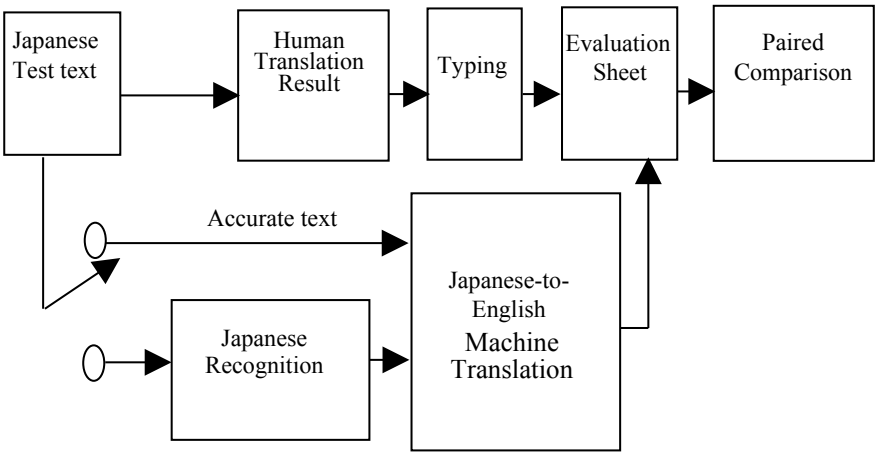

Fig. 6.1 Diagram of translation-paired comparison method

between. The pause time provided to humans for the translation was determined based on the questionnaire carried out in advance. The SLTA1 test set, which consists of 330 utterances from 23 conversations of a bilingual travel conversation database [3, 4] is used for the test. The SLTA1 test set is only used for evaluation purposes and not for the development of speech recognition and machine translation technologies.

Next, the handwritten translations are typed out and used in the following subjective comparison. In the proposed method, the translations made by the examinees and the output from the system are merged into evaluation sheets and are then compared by an evaluator, who is a native English speaker and also understands Japanese.

The evaluator comparatively ranks each utterance and repeats it for all test utterances. Each utterance information is shown in one of the evaluation sheets in the order of the Japanese test text and the two translation results, i.e., created by the examinees and by the system. The two translations are randomly ordered and presented to the evaluator to eliminate the evaluator's bias. The evaluator is asked to rank each utterance, following the steps illustrated in Fig. 6.2. The four ranks in Fig. 6.2 are the same as those used in [5]. The ranks A, B, C, and D indicate: (A) Perfect: no problems in both information and grammar; (B) Fair: easy-to-understand with some unimportant information missing or flawed grammar; (C) Acceptable: broken but understandable with effort; (D) Nonsense: important information has been translated incorrectly.

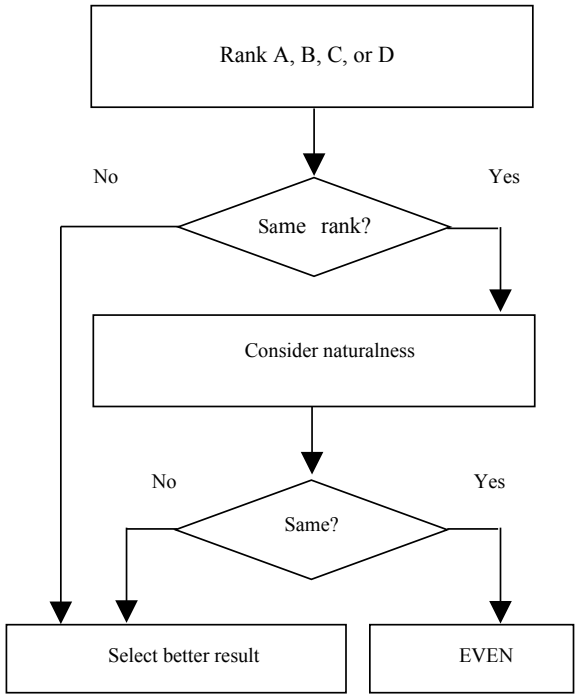

Fig. 6.2 Procedure of comparison for each utterance

6.3 Evaluation Result Using the Translation-Paired Comparison Method

Figure 6.3 shows the comparison results of the capabilities between machine translation and examinees described in Table 6.1. The dots in Fig. 6.3 indicate the measuring points. The MT input includes accurate transcriptions of utterances, which means there were no errors in speech recognition. The total number of examinees is thirty; five people in each score range from 300 s, 400 s, and up to 800 s. The horizontal axis in Fig. 6.3 represents the TOEIC scores of the examinees and the vertical axis shows the system winning rate (SWR) calculated by following equation:

$$SWR = \frac{N_{mt} + 0.5 \times N_{even}}{N_{total}} \tag{6.1}$$

where N_{TOTAL} denotes the total number of utterances in the test set, N_{MT} represents the number of "MT won" utterances, and N_{EVEN}, indicates the number of even (non-winner) utterances, i.e., no difference between the results of MT and humans. The SWR ranges from 0 to 1.0, indicating that the degree of capability of the MT system is relative to that of the examinee. An SWR of 0.5 means that the MT has the same capability as the human examinee.

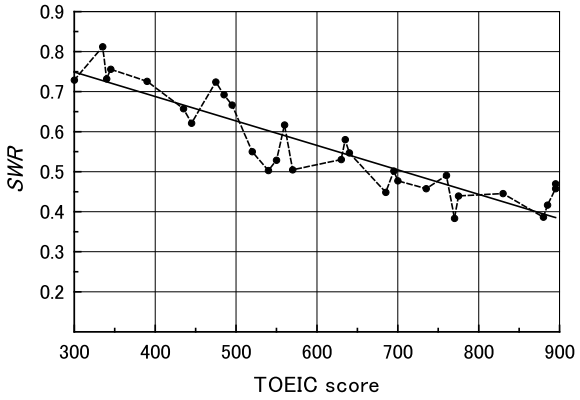

Fig. 6.3 Evaluation results using translation-paired comparison method

Table 6.1 Evaluation Parameters

The number of examinees	30 people
Japanese speech recognition (SPREC)	HMM acoustic model, n-gram language model
Machine translation (TDMT)	Rule-based
Speech translation (ATR-MATRIX)	SPREC + TDMT

Figure 6.3 also shows that the SWR of MT is greater than 0.5 at TOEIC scores of around 300–700, i.e., the MT system wins over humans with TOEIC scores between 300 and 700. On the other hand, examinees with TOEIC scores of around 800 win over MT. In order to precisely determine the balanced point, we used regression analysis which indicated that the capability-balanced area is just around 700. The straight line in Fig. 6.3 is the regression line which together with the measuring points indicate that the exact capability-balanced point is 708. We estimate this point as the system's TOEIC score. Consequently, the translation capability of the machine translation system equals that of the examinees at around a score of 700 points.

The Institute for International Business Communication (IIBC) in Japan provides a proficiency scale which shows the relationship between the TOEIC score and the communication proficiency. In this scale, five levels are defined. Level A is 860 points and up, B is between 730 and 860, and C is 470 and up. A TOEIC score of 700 of the system means that it is close to a Level B, surpassing C. A Level C communication proficiency means that speakers are capable of basic daily conversations and business communication in a limited domain. A Level B means that one is capable of proper communication in any situation and this is the level where the MT system is approaching towards.

The experimental result for MT combined with a speech recognition subsystem has been also reported in [2]. This system's TOEIC score is 548, where the speech recognition errors reduce the SWR from that of a single MT subsystem.

6.4 Error Analysis of the System's TOEIC Score

The SWR (Y_i) and TOEIC scores for the examinees (X_i) are assumed to satisfy the population regression equation:

$$Y_i = \beta_1 + \beta_2 X_i + \varepsilon_i \quad i = 1, 2, \ldots, n \tag{6.2}$$

where β_1 and β_2 are population regression coefficients. The error term (ε_i) is assumed to satisfy the following condition:

$$\begin{aligned} &\text{(a) } E(\varepsilon_i) = 0 \\ &\text{(b) } V(\varepsilon_i^2) = \sigma^2, \quad i = 1, 2, \ldots, n \\ &\text{(c) } Cov(\varepsilon_i, \varepsilon_j) = E(\varepsilon_i, \varepsilon_j) = 0 \quad \text{if } i \neq j \\ &\text{(d) } \varepsilon_i \cong 0 \end{aligned} \tag{6.3}$$

Under the above assumption, the standard deviation of the system's TOEIC score can be calculated by:

$$\sigma_t = \left|\frac{\sigma}{\beta_2}\right| \sqrt{\frac{1}{n} + \frac{(C_0 - \bar{X})^2}{\sum (X_i - \bar{X})^2}} \tag{6.4}$$

where n is the number of examinees, C_0 is the system's TOEIC score, and $\bar{X}$ is the average of the examinees' TOEIC scores. Equation (6.4) indicates that the minimum error is given when the system's TOEIC score is equivalent to the average of the examinees' TOEIC scores.

By using a t-distribution, the confidence interval (CI) of the system's TOEIC score with confidence coefficient $1 - \alpha$ is given by:

$$\begin{aligned} CI &= [C_0 - I, C_0 + I] \\ I &= \sigma_t \times t\left(\frac{a}{2}; n - 2\right) \end{aligned} \tag{6.5}$$

In Fig. 6.3, I is about 50 points when we employ 0.01 for the value of α Since we had planned to improve the system performance by 100-point notch in TOEIC score, the number of examinees (n) was set to 30. n can be set depending on the degree of improvement that the system aims for using Eq. (6.5).

In the previous evaluation using the translation-paired comparison method, one evaluator did the whole process shown in Fig. 6.2. We then performed another evaluation with the same setup but with a different evaluator. The difference between the two TOEIC scores was just 3.5 points in the case of Fig. 6.3.

6.5 Costs for the Translation-Paired Comparison Method

The translation-paired comparison method is an effective evaluation method as it clearly expresses the capability of a system with a TOEIC score. However, this method requires excessive evaluation costs compared with the traditional methods including subjective rank evaluation.

The collection of translations done by examinees of various TOEIC scores, is among one of the tasks that require a certain budget. As shown in Eqs. (6.4) and (6.5), n—the number of examinees—largely affects the confidence interval of the system's TOEIC score. Therefore, reduction in this number makes it difficult to obtain a reliable evaluation result.

Another cost-consuming task is subjective-paired comparison. Compared to the conventional evaluations i.e., simple rank evaluation method, the translation-paired comparison method uses a larger amount of labor because the evaluator must work on the n number of evaluation sheets for each test utterance. Even with an evaluator with high expertise, it takes more than two weeks to complete the steps shown in Fig. 6.2.

6.6 Automatic Method for TOEIC Evaluation

The basic idea of automating the translation-paired comparison method is to substitute the original human evaluation process i.e., the paired comparison in Fig. 6.1 with an evaluation metric including BLEU [1] and Translation Error Rate (TER).

Since the evaluation metric performs poorly with an individual utterance unit, we carry out the automation counting the whole test set as one unit. SWR for each examinee in the original method is substituted for the test set level metric to calculate the system's TOEIC score [6].

Figures 6.4 and 6.5 show the system's TOEIC scores of MT and MT with SR using BLEU and TER, which have been automatically estimated. In these figures, the vertical axes represent the system's TOEIC score, and the horizontal axes represent the number of references used in the evaluation metrics. Error bars in the figure show the confidence interval. In these figures white bars indicate the results in BLEU score, the gray bars indicate the results in TER.

The system's TOEIC score of MT using the automated method with 16 references marked 708 points, while the original translation-paired comparison method yielded a score of 705. With MT with SR, the proposed method scored 562 points and the original method was 546.

Considering the reductions in evaluation costs including labor and time, the performance of the proposed method is highly promising.

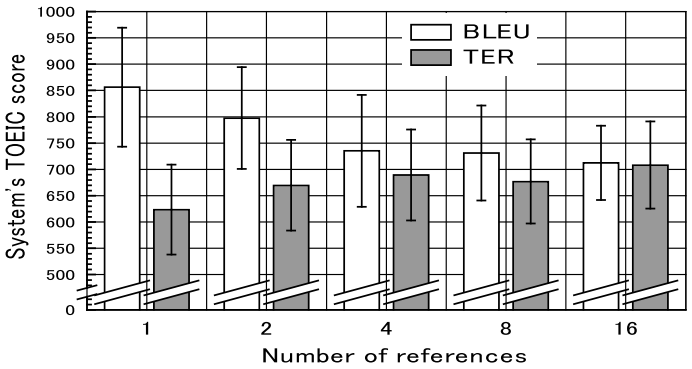

Fig. 6.4 Automatically-estimated system's TOEIC scores (MT)

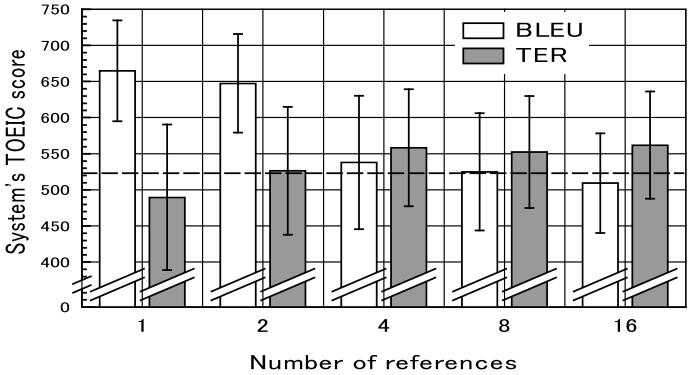

Fig. 6.5 Automatically-estimated system's TOEIC scores (MT with SR)

6.7 Conclusions

We proposed a method to measure the capability of a speech translation system which can be expressed in TOEIC scores. The errors in this method have also been analyzed. As a result of the error analysis, the number of examinees and its necessary TOEIC score distribution can be estimated depending on the project target. The method requires an enormous cost, which has also been explained. Back when this method was first proposed, the TOEIC score of the speech translation system as a whole was about 700. Considering the latest advance in the performance of speech recognition and translation components, the TOEIC score of the speech translation system is estimated to have risen significantly by the time this book is published in 2018. By the year 2020 when the Global Communication Plan would come to an end (further details are explained in Chap. 7), the system is expected to reach a score of 900 or higher which is said to be equivalent to that of a native

English speaker. Meanwhile, an automated method has also been proposed and showed promising results. However, in order to evaluate the system, a certain amount of foreign language learners and their test results are required, and to overcome this limitation and utilizing different tests are some of the future tasks that need to be undertaken.

References

1. Papineni, K., Roukos, S., Ward, T., Zhu, W.-J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 311–318 (2002)
2. Morimoto, T., Uratani, N., Takezawa, T., Furuse, O., Sobashima, Y., Iida, H., Nakamura, A., Sagisaka, Y., Higuchi, N., Yamazaki, Y.: A speech and language database for speech translation research. In: Proceedings of the Third International Conference on Spoken Language Processing (ICSLP 94), pp. 1791–1794 (1994)
3. Takezawa, T.: Building a bilingual travel conversation database for speech translation research. In: Proceedings of the 2nd International Workshop on East-Asian Language Resources and Evaluation Oriental COCOSDA Workshop '99, pp. 17–20 (1999)
4. Sumita, E.. et al.: Solutions to problems inherent in spoken language translation: the ATR-MATRIX approach. In: Proceeding of MT Summit, pp. 229–235 (1999)
5. Sugaya, F., Takezawa, T., Yokoo, A., Sagisaka, Y., Yamamoto, S.: Evaluation of the ATR-MATRIX speech translation system with a paired comparison method between the system and humans. In: Proceedings of International Conference on Spoken Language Processing (ICSLP), pp. 1105–1108 (2000)
6. Yasuda, K., Sugaya, F., Takezawa, T., Yamamoto, S., Yanagida, M.: Applications of automatic evaluation methods to measuring a capability of speech translation system. In: Proceedings of 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL), pp. 371–378 (2003)

Chapter 7
The Future of Speech-to-Speech Translation

Eiichiro Sumita

Abstract Since English is very different from Japanese in terms of grammar and vocabulary, it is not an easy task (http://www.effectivelanguagelearning.com/language-guide/language-difficulty) for a native English speaker to learn Japanese, and vice versa. Only a small number of native Japanese speakers can speak English, and the other way around is less likely. The part of the world where the Japanese language is used is basically limited to Japan. English, known as the "world language," is not commonly used in Japan and neither are other foreign languages. There is a clear communication barrier between the Japanese and non-natives, and the large need to overcome this became the motivation for researchers in Japan to initiate research on automatic speech-to-speech translation. We have so far briefly looked back on how the state-of-the-art technology was developed, but the following sections will explore some of the near future possibilities of the technology and how it will likely to be utilized in society before and after the year 2020.

7.1 The Future up to the Year 2020

As described in Chap. 6, the performance of speech translation can be measured by a scale, namely Test of English for International Communication (TOEIC). In the year 2000 when this measurement method was proposed, the system had already reached 600 points (out of 990). Considering the innovations in technology over the 20 years, by the year 2020—the final year of the Global Communication Plan

E. Sumita (✉)
Advanced Translation Technology Laboratory, Advanced Speech Translation Research and Development Promotion Center, National Institute of Information and Communications Technology, Kyoto, Japan
e-mail: eiichiro.sumita@nict.go.jp

Y. Kidawara et al. (eds.), *Speech-to-Speech Translation*, SpringerBriefs in Computer Science, https://doi.org/10.1007/978-981-15-0595-9_7

(GCP)—we can expect the system to achieve a very high score. It is of great significance that with the use of such high-precision systems, people who are not good at learning foreign languages can extend their abilities and the ripple effect that this has is huge.

The GCP aims to achieve the same precision for 10 languages[1] including Japanese and English, not only targeted for tourism, but for disaster resilience and medical care. Many types of hardware—megaphone type,[2] mobile type,[3] handsfree type,[4] standalone type[5]—which are capable of speech translation have been commercialized.

Figure 7.1 is one example of a handsfree device. Through field experiments on speech translation carried out in the medical field, it was learned that there are many situations in which the healthcare staff have their hands full—such as when providing care in a hospital—and that there was a great need for wearable speech translation devices that could be used without having to physically touch them. Technology was able to meet the demands by using small omnidirectional microphones to determine the differences between speakers. Accompanied by these hardware inventions, speech translation software is being widely accepted in various fields.

With the widespread of these speech translation systems throughout the country, by the 2020 Tokyo Olympic and Paralympic Games, Japan is expected to become one of the most "communication-friendly" countries in the world. The speech-to-speech translation technology which has been developed based on research that was initiated in Japan, has spread throughout the world and will return to Japan to witness its peak.

7.2 The Future Beyond the Year 2020

Among the many tasks that still need to be cleared, (1) high precision all-purpose translation, (2) simultaneous interpretation, and (3) context-based translation are the three that are assumed to be unfinished by the year 2020. In other words, there is still plenty of room for further research and development to be done in the field of automatic speech-to-speech translation.

[1]Japanese, English, Chinese, Korean, Indonesian, Thai, Vietnamese, Myanmar, French, and Spanish.

[2]http://news.panasonic.com/global/topics/2016/45751.html.

[3]http://www.nec.com/en/press/201710/global_20171002_02.html.

[4]http://www.fujitsu.com/global/about/resources/news/press-releases/2017/0919-01.html.

[5]http://logbar.jp/en/index.html.

Fig. 7.1 A wearable, handsfree speech translation device

7.2.1 Future Tasks: High Precision All-Purpose Translation System

The current technology relies heavily on the corpus. The performance improves as the amount of corpus increases. Conversely, high performance cannot be expected with an insufficient amount of corpus. A race between organizations that pursue research and development has begun to see who can collect more data, but the inconsistent accuracies of many translation services—due to limitations in language or domain—clearly state that the collection method needs to be reexamined. Thus, a high precision all-purpose translation system has yet to be realized.

The bilingual corpora are dispersed among different organizations, e.g., private companies, and local governments. As shown in Fig. 7.2, if all of these bilingual corpora are accumulated into one public sector and deep learning is applied, the translation accuracy is expected to rise significantly.

For example, in Japan, a budget of 200 billion yen is being spent on translation each year. From this, we can estimate that the number of sentences being translated per year is about 500 million, and in 10 years it will be 5 billion sentences. The precision that the system will achieve using such large-scale corpus will surpass our expectations and can be used in every field in need of translation services. The Government of Japan has named this mechanism "Hon'yaku Bank," and is promoting development of a machine translation system of the people, by the people, and for the people.

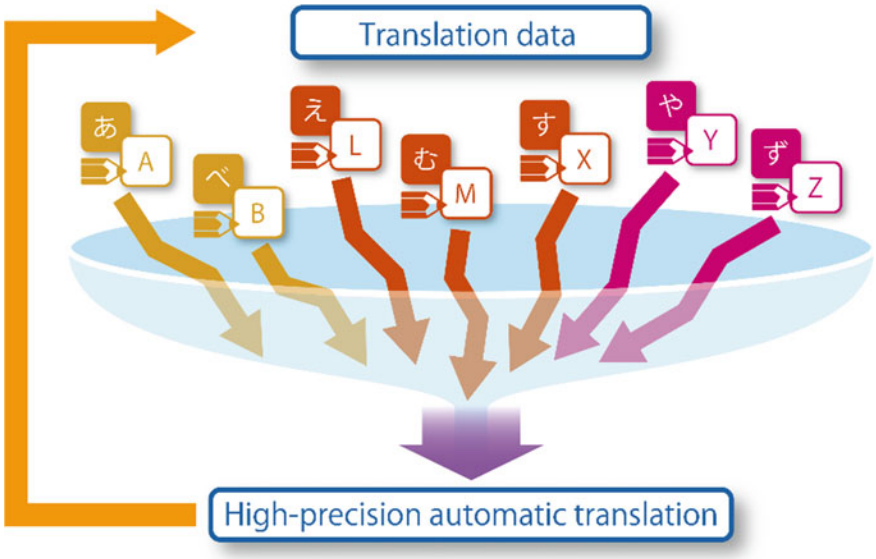

Fig. 7.2 Concept of "Hon'yaku Bank" ("Hon'yaku" is a Japanese word for "translation")

7.2.2 Future Tasks: Simultaneous Interpretation

The current speech translation system begins the translation process after one has finished speaking. The completion of the process will be largely delayed when the input is long. In order to avoid such delay, the system needs to start processing before the utterance reaches its end.

Simultaneous interpretation from Japanese into English is especially challenging due to the difference in word order. The Japanese word order is Subject (S)-Object (O)-Verb (V), and V—the most important part of speech—is not revealed until the end of a sentence. English is S-V-O, and V comes after the subject. In order to translate the V to English from Japanese, one must wait until the Japanese sentence is completed. An interpreter would however, start translating without waiting for a sentence to finish, which means that the translation is processed with a prediction and that the interpreter must go back to make corrections when the translation turns out to be wrong.

In such way, simultaneous interpretation is difficult even for humans, and to realize simultaneous interpretation using a computer, it requires a great deal of time and a huge breakthrough in technology.

On the other hand, a few fundamental research studies [1] have been initiated and the current system is capable of accommodating the development strategy of utilizing the system as a supportive tool for international (i.e., multilingual) conferences and gradually improving the performance.

7.2.3 Future Tasks: Context-Based Translation

The current speech-to-speech translation system and its elemental technologies only process translation sentence by sentence, without any consideration of the preceding sentences.

Studies on context have also been carried out for many years, however, the applicability and processing speed are some of the issues that still remain. Translation is often performed by determining the surrounding words. If translating sentence by sentence would be practical enough, context may not have to be taken into account.

On the other hand, the latest deep learning techniques have shown significant results in context understanding. Some of the typical issues in context-based translation are detecting the omission of the subject—which frequently occurs in Japanese—and identifying them to perform accurate translation, but in this regard, it has been reported [2] that deep learning has effectively improved the performance compared to the conventional methods. The authors expect with great anticipation that this will further advance the system to accommodate context-based translation.

References

1. Wang, X., Finch, A., Utiyama, M., Sumita, M.: A prototype automatic simultaneous interpretation system. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations, pp. 30–34, Osaka, Japan, 11–17 Dec 2016
2. Iida, R., Torisawa, K., Hashimoto, C., Oh, J.-H., Kloetzer, J.: Intra-sentential zero anaphora resolution using subject sharing recognition. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2015), pp. 2179–2189, Lisbon, Portugal, September 2015