A study on Mongolian text-to-speech system based on deep neural network

March 2022

Doctor of Engineering

Zolzaya Byambadorj

Department of Information Science and Intelligent Systems
Graduate School of Advanced Technology and Science
Tokushima University

Abstract

There are about 7,000 languages spoken today in the world. However, most natural language processing and speech processing studies have been conducted for high resource languages such as English, Japanese and Mandarin. Preparing large amounts of training data is expensive and time-consuming, which creates a significant hurdle when developing some systems for the world's many, less widely spoken languages. Mongolian is one of these low-resource languages. We proposed to build a text-to-speech system (TTS, also called speech synthesis) for the low resource Mongolian language. We present two studies within this TTS system, "*text normalization*" and "*speech synthesis*," on the Mongolian language with limited training data. TTS system converts written text into machine-generated synthetic speech. One of the biggest challenges to developing a TTS system for a new language is converting transcripts into a real "spoken" form, the exact words that the speaker said. This is an important preprocessing for TTS systems known as text normalization. In other words, text normalization is transforming text into a standard form and is an essential part of the speech synthesis system. Later it also became important for processing social media text because of the rapid expansion in user-generated content on social media sites. As the use of social media grows rapidly, there is no doubt that the TTS system will need to generate speech from social media text. Therefore, we were more interested in social media text normalization. Thus, this thesis consists of two main parts, text normalization and speech synthesis. We experimentally demonstrated how to improve the output of the model used for each using a small amount of training data. The followings are brief descriptions of each part.

Text normalization: The huge increase in social media use in recent years has resulted in new forms of social interaction, changing our daily lives. Social media websites are a rich source of text data, but the processing and analysis of social media text is a challenging task because written social media messages are usually informal and 'noisy'. Due to increasing contact between people from different cultures as a result of globalization, there has also been an

increase in the use of the Latin alphabet, and as a result a large amount of transliterated text is being used on social media. Although there is a standard for the use of Latin letters in the language, the public does not generally observe it when writing on social media. Therefore, social media text also contains many noisy, transliterated words. For example, many people who speak Mongolian are using the Latin alphabet to write Mongolian words on social media, instead of using the Cyrillic alphabet. These messages are informal and 'noisy' however, because everyone uses their own judgement as to which Latin letters should be substituted for particular Cyrillic letters, since there are 35 letters in the Mongolian Cyrillic alphabet, versus 26 letters in the modern Latin alphabet (not counting letters with diacritical marks such as accents, umlauts, etc.). In most research on noisy text normalization, both the source text and target text are in the same language. In other words, the alphabets used in the source and target texts are the same. Text normalization is difficult to perform with noisy text even when it is not transliterated. In this thesis, our first goal is to convert noisy, transliterated text into formal writing in a different alphabet. Therefore, it poses more challenges in the text normalization task. We propose a variety of character level sequence-to-sequence (seq2seq) models for normalizing noisy, transliterated text written in Latin script into Mongolian Cyrillic script, for scenarios in which there is a limited amount of training data available. When there is a limited amount of training data, and the rules for writing noisy, transliterated text are not limited, we encounter a difficult challenge when attempting to normalize out-of-vocabulary (OOV) words. Therefore, we applied performance enhancement methods, which included various beam search strategies, N-gram-based context adoption, edit distance-based correction and dictionary-based checking, in novel ways to two basic seq2seq models. We experimentally evaluated these two basic models as well as fourteen enhanced seq2seq models, and compared their noisy text normalization performance with that of a transliteration model and a conventional statistical machine translation (SMT) model. The proposed seq2seq models improved the robustness of the basic seq2seq models for normalizing OOV words, and most of our models achieved higher normalization performance than the conventional method.

Speech synthesis: Deep learning techniques are currently being applied in automated TTS systems, resulting in significant improvements in performance. These methods require large amounts of text-speech pair data for model training however, and collecting this data is costly.

Tacotron 2 we used, a state-of-the-art end-to-end speech synthesis system, requires more than 10 hours of training data to produce good synthesized speech. Therefore, our second goal is to build a single-speaker TTS system containing both a spectrogram prediction network and a neural vocoder for the target Mongolian language, using only 30 minutes of target Mongolian language text-speech paired data for training. We evaluate three methods for training the spectrogram prediction models of our TTS system, which produce mel-spectrograms from the input phoneme sequence; (1) cross-lingual transfer learning, (2) data augmentation, and (3) a combination of the previous two methods. In the cross-lingual transfer learning method, we used two high-resource language datasets, English (24 hours) and Japanese (10 hours). We also used 30 minutes of target language data for training in all three methods, and for generating the augmented data used for training in methods (2) and (3) mentioned above. We found that using both cross-lingual transfer learning and augmented data during training resulted in the most natural synthesized target speech output. We also compare single-speaker and multi-speaker training methods, using sequential and simultaneous training, respectively. The multi-speaker models were found to be more effective for constructing a single-speaker, low-resource TTS model. In addition, we trained two Parallel WaveGAN (PWG) neural vocoders, one using 13 hours of our augmented data with 30 minutes of target language data and one using the entire 12 hours of the original target language dataset. Our subjective AB preference test indicated that the neural vocoder trained with augmented data achieved almost the same perceived speech quality as the vocoder trained with the entire target language dataset. We found that our proposed TTS system consisting of a spectrogram prediction network and a PWG neural vocoder was able to achieve reasonable performance using only 30 minutes of target language training data. We also found that by using 3 hours of target language data, for training the model and for generating augmented data, our proposed TTS model was able to achieve performance very similar to that of the baseline model, which was trained with 12 hours of target language data.

Keywords

Text normalization, noisy text, transliterated text, language model, seq2seq model, character conversion, speech synthesis, text to speech, transfer learning, data augmentation, low resource language

Acknowledgement

First of all, I would like to express my deep gratitude to my supervisor Professor Norihide Kiatoka of the Department of Computer Science and Engineering at the Toyohashi University of Technology, for the continuous support of my doctoral study. His valuable advice, motivation, and guidance helped me throughout my research and in writing this thesis. I am very grateful to him for learning a lot from him in the past three years. I could not have completed this study without his guidance and advice.

My grateful thank is also extended to Associate Professor Ryota Nishimura of the Department of Information Science and Intelligent Systems at the Tokushima University and Associate Professor Kengo Ohta of the Department of Creative Technology Engineering at the National Institute of Technology, Anan College, for their advice, support and assistance in keeping my progress on schedule.

I would also like to thank Professor Altangerel Ayush and Professor Khishigjargal Gonchigsumlaa of the School of Information and Communication Technology, Mongolian University of Science and Technology. They helped me to find a supervisor and gave me this wonderful opportunity to study for a Ph.D. in Japan.

I would also like to express my gratitude to Ms. Meiko Fukuda, who always helped me while living and studying in Japan.

I would like to express my sincere gratitude to the organizations implementing the Higher Engineering Education Development Project. I received a great opportunity to study doctoral program in Japan and full financial support from the project.

Finally, I would like to give special thanks to my family for their continuous support and understanding when undertaking my research and writing my project.

List of Figures

List of Tables

Contents

Chapter 1: Introduction

1.1 Background and aims

TTS is an assistive technology that can be used in many ways. For example, there are TTS tools available for nearly every digital device, including computers, smartphones, and tablets. In addition, this software is also incorporated into the websites. TTS tool reads digital text aloud. As a result, it can help people with hearing impairments, disabilities, and aged citizens to better understand the content. Besides, it is also helpful for kids who struggle with reading. However, popular speech services only support a few languages because these few languages have sufficient resources, such as the large amount of speech data used to train the TTS model. Xu et al. (2020) mentioned that although there are more than 6,000 languages in the world, most languages lack speech training data, while Chen et al. (2019) said that laborious data collection remains difficult for at least 95% of the languages over the world. Magueresse et al. (2020) also notes that most of the natural language processing (NLP) research focuses on 20 of the 7,000 languages of the world, and the vast majority of languages remain not well studied. These languages are often referred to as low-resource languages. Therefore, data scarcity is a significant challenge when building a TTS system for extremely low-resource languages. Investigating methods to resolve data sparsity is very helpful for low-resource languages. Mongolian is one of the low-resource languages.

In this thesis, we proposed to build a TTS system for the low resource Mongolian language when a limited amount of training data is available. We present two studies, "*text normalization*" and "*speech synthesis*," related to our proposed TTS system. Although we have not yet conducted an experiment that combines the two parts, we have tested them separately and investigated the best method for each case. An overview of the proposed TTS system with the whole process is illustrated in Figure 1.1.

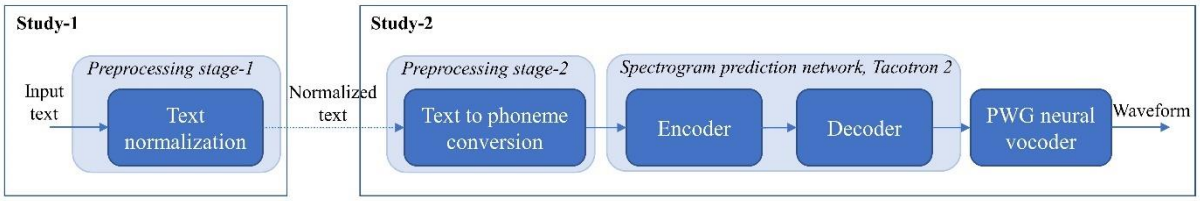

Figure 1.1. Overview of the TTS system with the whole process.

TTS system converts written text into machine-generated synthetic speech. The TTS system consists of two main components, a spectrogram prediction network and a vocoder. In other words, to generate a speech signal, we need two main steps; the first step is generating a mel-spectrogram from the input phoneme sequence using the spectrogram prediction network, and the second step is generating the waveform from the mel-spectrogram using the vocoder. We used Tacotron 2 model as the spectrogram prediction network, while Parallel WaveGan (PWG) neural vocoder was used to generate waveform from the mel-spectrogram in our TTS system. In this study called "*speech synthesis*," we investigated various methods to train the TTS system consisting of both components, with only 30 minutes of Mongolian speech data. The second block (Study-2) in Figure 1.1 shows a flow diagram of the speech synthesis part.

In more detail, there are two sub-networks, encoder and decoder, in the spectrogram prediction network. First of all, there is a preprocessing stage that converts an input text into a phoneme sequence. The encoder network extracts out a hidden feature representation from this phoneme sequence. The output of the encoder is then fed into the decoder network, which generates the mel-spectrogram from it. However, there is one more important preprocessing step, which is text normalization. It takes input text and analyzes and organizes the input into a manageable list of words because the input text may contain symbols, numbers, abbreviations, acronyms, etc. In other words, text normalization is transforming text into a standard form. For example, the number "20" should be converted into its standard form "twenty." Therefore, text normalization provides that these non-standard words are pronounced easily by a TTS system. Normalization of the input text into its actual pronunciation depends on the context of its use. Therefore, it is one of the biggest challenges to developing a TTS system for a new language. In this study called "*text normalization*", we address the issue of

noisy, transliterated text normalization, which has recently attracted the attention of many researchers. If text is challenging to attempt to derive meaning from the words by software, it means noisy text. Text in chats, blogs, and SMS is very noisy because abbreviations and slang are widely used. Transliteration is the process of converting texts from one script to another based on phonetic similarity. For example, in the normalization task, transliterated text written in the Latin alphabet should be normalized into a canonical form that uses a different set of characters. Therefore, the normalization of noisy, transliterated text is more complicated than both non-standard text and noisy text in its native alphabet. The first block (Study-1) in Figure 1.1 shows this text normalization part of our proposed TTS system.

In general, because of the huge increase in social media use in recent years, our goal is to build a TTS system that can generate speech from any text such as canonical and noisy, transliterated text. Section 1.2 describes the noisy, transliterated Mongolian words and a brief overview of our methods to address them. Section 1.3 explains a brief overview of how we addressed the problem of data scarcity to train recent end-to-end neural models, which require a large amount of training data.

1.2 Text normalization

Social media websites such as Facebook, Twitter, Instagram and Snapchat are a rich source of text data, but the processing and analysis of social media text is a challenging task because written social media messages (chat, discussion, comments, etc.) are usually informal and 'noisy', i.e., the writers use non-standard grammar and abbreviations, acronyms, phonetic substitutions of words with numbers or letters (LOL, CUL8R, etc.), slang, emojis, and many misspelled words. Social media text also contains many transliterated words, which are the result of converting words from the letters used in the source language into the letters used by another language, based on phonetics. For example, the English word 'Twitter' becomes 'ツイッター' (tsuittaa) in Japanese.

In Mongolia, there are currently two writing systems in use, traditional Mongolian script and Mongolian Cyrillic, the latter of which is the country's official alphabet. But even though

the official writing system is Mongolian Cyrillic script, many Mongolians are using the Latin alphabet when posting information in Mongolian on social media sites, such as Facebook and Twitter, and when using mobile devices to send text messages. A standard for the transliteration of the Mongolian Cyrillic alphabet into the Latin alphabet (MNS 5217:2003) was approved in 2003, due to the increasing use of the Latin alphabet resulting from the proliferation of modern technology and the Internet. In 2012, the transliteration standard was revised (MNS 5217:2012). Both old and revised standards can be found in Appendix A and B, respectively. Although the Mongolian language has this standardized method for transliteration of Mongolian Cyrillic into Latin characters, the public does not generally observe it when writing on social media. As a result, transliterated Mongolian words are written in several different ways when using the Latin alphabet. For example, the given name "Мөнхсүх" can be transliterated as Munkhsukh, Munkhsvkh, Monkhsukh, Monhsuh, Munhsuh, Munxsvx, and Monhsvh. Before it became possible to use the Mongolian Cyrillic alphabet for mobile phone text messaging, people could only send short message service (SMS) messages using the Latin alphabet. Although it is now possible to write SMS text in the Mongolian Cyrillic alphabet, use of the Latin alphabet is still common because it is easier and faster to use. Guruuchin (2018) conducted a small survey to determine why people use the Latin alphabet when they write in Mongolian. A total of 74 Facebook users, divided into three age groups (16-26, 27-40 and 41-55), participated in the survey. The reasons reported for using the Latin alphabet when writing in Mongolian were as follows:

- It is faster to write when using the Latin alphabet. (61.7%)
- When using the Latin alphabet, there are no spelling rules. (60.9%)
- Computers and mobile phones have Latin alphabet keyboards. (56.4%)

When asked the question, "Do you know the standard method of transliterating the Mongolian Cyrillic alphabet into the Latin alphabet?", 85% of the participants said they did not know the standard method, while 15% said they knew it. Of those who said they knew the standard method, 10% said they use it, while the other 5% said they do not use it. The author concluded that people use the Latin alphabet according to their own judgement, and that the

main reason for using the Latin alphabet was its use on the keyboards of computers and mobile phones.

Processing social media text is one of the key targets of NLP, thus there has been much research in recent years focusing on social media. However, there has been a lack of research in this area focusing on the Mongolian language. In fact, this is the first study on text normalization for Mongolian.

Originally, text normalization was used to convert words that appeared in non-standard formats, such as numbers, dates, acronyms and abbreviations, into standard forms (Zhang et al. 2019; Huang et al. 2020). But later, text normalization was also applied to the conversion of informal text on social media into formal text. Table 1.1 shows examples of these two types of text normalization.

Table 1.1. Examples of text normalization

Category	Non-standard form	Standard form	Informal text	Formal text
Date	2020	'two thousand twenty' or 'twenty twenty'	c u	see you
Money	$10	'ten dollars'	cooooooolllll	Cool
Time	17:10	'five ten' or 'ten after five' or 'ten past five'	BRB	I'll be right back

In most research on noisy text normalization, both the source text and target text are in the same language. Our goal in this study, however, is to convert noisy, transliterated text into formal writing in a different alphabet. In other words, the alphabets used in the source and target texts are different (the Latin and Mongolian Cyrillic alphabets, respectively). For convenience, in the rest of this thesis we will use the term 'Cyrillic' to refer to the Mongolian Cyrillic alphabet. The aim of this text normalization study is the exploration of variants of character-level seq2seq models which are able to outperform conventional methods when only a limited amount of training data is available. We achieve this by combining various methods

(seq2seq models which use different beam search strategies, N-gram based context adoption, edit distance-based correction, and dictionary-based checking) in novel ways, specifically designed to overcome the challenges presented by OOV words.

1.3 Speech synthesis

Speech synthesis is the computer-based creation of artificial speech from normal language text. As for the speech synthesis, there has also been a lack of research in this area focusing on the Mongolian language. However, there are a few related research works for the Mongolian language, and most of them used traditional methods (Davaatsagaan and Paliwal 2008; Zhao et al. 2014). There are two types of Mongolian script, as we mentioned in Section 1.2. Some authors of the related works used text-speech paired data containing transcripts of utterances written in the traditional Mongolian script, while others used data containing transcripts written in the Mongolian Cyrillic script. Liu et al. (2017) built Mongolian TTS system based on the deep neural network (DNN). They transformed the traditional Mongolian script into their corresponding Latin transcriptions, then the statistic-based Mongolian grapheme to phoneme (G2P) conversion method was used to generate Mongolian phoneme sequence. They compared the DNN-based system with the conventional HMM-based system under the same corpus, and their results showed that the performance of the DNN-based system was better than the HMM-based system. Even if their system synthesized clear speech, it could not give a natural-sounding synthesized speech. In general, these traditional methods produce the audio to sound less natural than human speech. Deep learning techniques are now widely used in TTS systems due to their ability to generate higher quality synthesized speech than traditional methods. For example, recent end-to-end neural models such as Tacotron (Wang et al. 2017), Tacotron 2 (Shen et al. 2018), Deep Voice 3 (Ping et al. 2018) and Char2Wav (Sotelo et al. 2017) are all able to generate natural-sounding speech. There is a related work (Li et al. 2018) for the Mongolian language, which used the Tacotron model for the Mongolian speech synthesis system. They built a large-scale dataset consisting of about 17 hours of recordings for training. Their experimental results showed that the proposed end-to-end Tacotron model performed

better than HMM-based and DNN-based models. Therefore, in this study, we use the deep-learning method to build our proposed TTS model.

A deep neural network is usually trained using a corpus of several hours of recorded speech from a single speaker. Hence, it is called a single speaker TTS model. In addition to a single-speaker TTS model, it has recently been extended to support multi-speaker voices, speech synthesis in more than one language, and target language speech synthesis using speakers of different languages (cross-lingual TTS). But this study aims to build a single-speaker Mongolian TTS system using Tacotron 2, a state-of-the-art end-to-end speech synthesis system, when a very limited amount of target language training data is available. However, the end-to-end neural models we mentioned above require a large amount of paired text-speech data for training, as well as substantial processing power. For example, Chung et al. (2019) found that the Tacotron model requires more than 10 hours of training data to produce good, synthesized speech. But collecting large amounts of speech data is expensive and time-consuming, which creates a significant hurdle when developing TTS systems for the world's many, less widely spoken languages. Therefore, our main issue in this study is building a TTS system when only a small amount of target data is available.

In this study, we used only 30 minutes of the target language training data to build our proposed TTS system containing both a spectrogram prediction network (Tacotron 2) and a neural vocoder (PWG) for the target language. The spectrogram prediction model produces mel-spectrograms from the input phoneme sequence, while the neural vocoder generates the waveform conditioned on mel-spectrogram. However, it is impossible to synthesize intelligible speech using the model trained with only 30 minutes of target language data from scratch. Therefore, we tested following three approaches for training the spectrogram prediction models of our TTS system to solve the problem of data scarcity.

1. Cross-lingual transfer learning method
2. Data augmentation method
3. A combination of the previous two methods

The cross-lingual transfer learning method means that the knowledge learned from a large amount of data is transferred and a pre-trained model is adapted with speech data in another

language. Therefore, in the cross-lingual transfer learning method, we used two high-resource language datasets, English (24 hours) and Japanese (10 hours). The second method we used to solve the problem of data scarcity is data augmentation, which is used to artificially create new training data from existing training data. We generated synthetic data from the 30 minutes of the original target language data to solve the limited target language data issue and increased the amount of target language training data 27 times the size of the original dataset. In addition to using the cross-lingual transfer learning and data augmentation methods separately, we also combined these two methods to improve the performance of TTS model. Besides, we also used the augmented data to train the neural vocoder.

As a result, our proposed TTS system, consisting of a spectrogram prediction network and a PWG neural vocoder, was able to achieve reasonable performance using only 30 minutes of original target language training data.

1.4 Thesis structure

The thesis is organized as follows. As mentioned before, this thesis consists of two main parts, text normalization and speech synthesis, which are components of our proposed TTS system. Chapter 2 deals with the text normalization part of our TTS system, while Chapter 3 covers the speech synthesis part. Chapters corresponding to both parts/studies contain the related work, the architecture of all models used to conduct our experiments and the datasets used. Besides, we also describe each of the various methods we tested when building our proposed system for each study and discuss each study's experimental results in Chapter 2 and Chapter 3, respectively. Finally, Chapter 4 includes our general conclusion and future work.

Chapter 2: Text normalization of Mongolian text transliterated to Arabic

2.1 Literature review

Due to the increase in the use of noisy text on social media sites, noisy text normalization has become a major research topic in the field of NLP. Aw et al. (2006) used a phrase-based statistical model for short messaging service (SMS) text normalization. Vilariño et al. (2012) used a statistical bilingual dictionary, constructed using the IBM-4 model, to normalize SMS texts. Saloot et al. (2014) developed an unsupervised normalization system with two phases, candidate generation and candidate selection, for noisy text normalization. Six methods were used to generate candidates; one-edit distance lexical generation, phonemic generation, blending the previous two methods, two-edit distance lexical generation, dictionary translation and the use of heuristic rules. A language model probability score was then used to select the most appropriate candidate. Kaur and Mann (2016) developed a hybrid approach consisting of SMT and direct mapping, to transform non-standard text into standard text.

More recently, several neural methods for machine translation have been proposed which can also be used for text normalization, such as Kalchbrenner and Blunsom (2013), Cho et al. (2014), Sutskever et al. (2014), Bahdanau et al. (2014) and Luong et al. (2015). Ikeda et al. (2016) used a character-level encoder-decoder model for normalizing noisy Japanese text, and also built a synthetic noisy text database with pre-defined rules for data augmentation. They compared their neural network model with a rule-based method, as well as with using Conditional Random Fields (CRF). Lusetti et al. (2018) normalized Swiss German WhatsApp messages using a neural network model by integrating a language model into a character-level neural model. They then compared the performance of their model with that of a state-of-the-art character-level SMT method. Lourentzou et al. (2019) used a hybrid seq2seq model which

consisted of two, nested encoder-decoder architectures, namely, word-level and character-level seq2seq models. When an unknown symbol is encountered by the word-level seq2seq model, the character-level seq2seq model is used to normalize these OOV words. This hybrid model has achieved the best text normalization performance so far among neural models, but its performance is still below that of some conventional methods. Mager et al. (2019) proposed the use of a novel auxiliary text normalization task for seq2seq neural architectures, which improved the performance of the base seq2seq model by up to 5%. This increase in performance closed the gap between SMT and neural approaches for low-resource text normalization. Tursun and Cakici (2017) normalized noisy Uyghur text containing unsystematic use of the Latin alphabet into text written in the Common Turkic Alphabet (CTA), and compared the performance of a noisy channel model and a neural encoder-decoder model as normalization methods, using both synthetic and authentic data. They selected the character-based solution in the encoder-decoder model, but chose the word-based solution for the noisy channel model. Mandal and Nanmaran (2018) normalized noisy, transliterated Bengali words written in the Latin alphabet into words in native Bengali script using a seq2seq model and the Levenshtein distance algorithm (Levenshtein 1966). The last study cited is similar to our research, because transliterated text written in the Latin alphabet was normalized into a canonical form which uses a different set of characters. However, while we seek to directly convert noisy, transliterated text into a canonical form, they first converted the noisy, transliterated text into standardized, transliterated text as accurately as possible. Then Levenshtein distance was performed to find the most appropriate canonical word, using a dictionary which consisted of canonical words and the corresponding words in their standardized, transliterated form.

2.2 Text normalization challenge

In this section, we describe the challenges of normalizing noisy, transliterated Mongolian words. Non-standard forms generated by writers on social media include words that are misspelled, abbreviated or shortened, as well as the use of phonetic substitution, acronyms and slang. We face all these problems in our project and, in addition, we must deal with the

widespread use of the Latin alphabet. The challenges encountered when normalizing noisy, transliterated text into standard Mongolian can be described as follows:

- The Mongolian Cyrillic alphabet has 35 letters, and 17 of them have more than one transliteration alternative in the Latin alphabet. Table 2.1 shows the possible alternative Latin characters for Cyrillic letters, based on our training data (Table 2.8 in Section 2.3.1), but it does not consider Latin alternatives for misspelled text (swapping two letters or replacing letters with others located nearby on the keyboard) or abbreviations.
- Writers generally choose from among these alternatives based on phonetics. However, the phonetics of some Latin alternatives do not exactly match the phonetics of the Cyrillic characters, so they choose the alternative which they feel sounds more similar.
- Some substitutions that were approved under the previous standard (MNS 5217:2003) are still in use. For example, the standard transcriptions of 'ө' and 'ү' are 'ö' and 'ü,' respectively, in the MNS 5217:2012 standard. But depreciated transcriptions are 'o' and 'u,' respectively. The standard transcription of 'ц' is 'ts' in the MNS 5217:2012 standard, but 'c' in the MNS 5217:2003 standard, etc.
- In addition, the phonetics of some Latin alternatives are very different from the phonetics of the original Cyrillic characters, but since the appearance of a Latin character is the same or similar to a particular character in the Cyrillic alphabet, users may choose it over another alternative. For example, the Cyrillic letters in rows 6, 7, 8, 9, and 11 of Table 2.1 have the same appearance as some Latin letters. Although 'ү' and 'v' do not look exactly the same, some writers use 'v' as a Latin alternative because the two characters look somewhat similar. Table 2.2 shows some examples of these kinds of alternatives.

Table 2.1. Latin alphabet alternatives for Cyrillic letters

#	Cyrillic letter	Standard transcription MNS 5217:2012	Latin alternatives
1	в	v	v, w, b
2	е	ye	ye, e, y, i
3	ё	yo	yo, e, y, i
4	к	k	k, c
5	ө	ö	o, u
6	р	r	r, p
7	с	s	s, c
8	у	u	u, y
9	ү	ü	u, y, v
10	ф	f	f, p
11	х	kh	kh, h, x
12	ц	ts	ts, c
13	ч	ch	ch, ts, c, j
14	ы	y	i, y, ii
15	ь	i	i, e
16	ю	yu	yu, y
17	я	ya	ya, y

Table 2.2. Examples of Latin letters used in transliterated words that look similar to Cyrillic letters

Word in Cyrillic script	Incorrect transliteration into Latin script	Standard transliteration into Latin script (meaning)
ө**р**гө**х**	o**p**go**x**	örgökh (to lift)
зал**уу**	zal**yy**	zaluu (guy)
х**ү**сье	h**v**sey	khüsiye (good luck)

- Another problem is that a single Latin letter is used to transliterate multiple different Cyrillic letters when using the MNS 5217:2012 standard (see Table 2.3). For example, the four Cyrillic letters ‘и,’ ‘й,’ ‘ъ’ and ‘ь’ are transliterated into the Latin letter ‘i,’ while the two letters ‘ш’ and ‘щ’ are both transliterated into the Latin letters ‘sh.’ This situation also makes it difficult to normalize noisy, transliterated, Mongolian text into Cyrillic script.

Table 2.3. Common Latin alphabet alternatives to the standard transcriptions of Cyrillic characters

#	Cyrillic letter	Standard transcription MNS 5217:2012	Latin alternatives
1	и	i	i
2	й	i	i
3	ъ	i	i
4	ь	i	i, e

- Transcriptions of some Cyrillic letters consist of two Latin letters, according to the MNS 5217:2012 standard (see Table 2.4). These kinds of transcriptions also make normalizing noisy, transliterated text into Cyrillic script challenging.

Table 2.4. Standard transcriptions of particular Cyrillic letters and common Latin alphabet alternatives

#	Cyrillic letter	Standard transcription MNS 5217:2012	Latin alternatives
1	е	ye	ye, e, y, i
2	ё	yo	yo, e, y, i
3	х	kh	h, kh, x
4	ц	ts	c, ts
5	ч	ch	ch, ts, c, j
6	ш	sh	sh
7	щ	sh	sh
8	ю	yu	yu, y
9	я	ya	ya, y

- Tables 2.5 and 2.6 show examples of Cyrillic words with alternative and ambiguous transliterations contained in our training data (Table 2.8 in Section 2.3.1), illustrating how words written in Cyrillic script can be transliterated into many non-standard Latin alphabet forms. For example, the two Cyrillic words shown in the last row of Table 2.5 (байдаг юм) can be transliterated into any of the ten, single-word, noisy non-standard forms shown in Latin script, in addition to the eleventh, two-word, standard transliteration. Therefore, when performing normalization, one-to-many mapping (1 : N) is used. Conversely, the standard transliteration ‘utsaar,’ shown in Table 2.6, can be the transliteration of two different Cyrillic words. This occurs because in the letter ‘ц’ in the Cyrillic alphabet is transliterated as ‘ts’ in Latin characters, while the Cyrillic characters ‘т’ and ‘с’ are transliterated separately as ‘t’ and ‘s,’ making it very difficult to determine whether the letters ‘t’ and ‘s’ in transcriptions are being used separately or together.

Table 2.5. Examples of normalizations of noisy, transliterated Mongolian words

Phonetically transliterated word in Latin script	Standard transliteration in Latin script (meaning)	Word in Cyrillic script	Type of alignment
hucie, husey, husie, husii, hvsey, hvsie, hvsyaa, xusie, xvsie, xvsii	khüsiye (good luck)	хүсье	1 : 1
bayarllaa, bayralaa, bayrallaa, bayrlala, bayrlalaa, bayrlalaaa, bayrlla, bayrllaa, byrllaa	bayarlalaa (thank you)	баярлалаа	1 : 1
zovlogoo, zovolgoo, zowlogoo, zuvluguu, zuvulgoo, zuvulguu, zuwluguu, zuwulguu	zövlögöö (advise)	зөвлөгөө	1 : 1
baidgiim, baidiim, baidin, bdagiin, bdgiin, bdi, bdiim, bdiin, bdim, bdy	baidag yum (there is)	байдаг юм	1 : N

Table 2.6. Examples of ambiguous transcriptions

Phonetically transliterated word in Latin script	Standard transliteration into Latin script	Possibility 1 in Cyrillic script (meaning)	Possibility 2 in Cyrillic script (meaning)
utsar	utsaar/utsaar	утсаар (by phone)	уцаар (huff)
uul	uul/üül	уул (mountain)	үүл (cloud)

If the amount of training data is small, and the rules for writing noisy, transliterated text are not limited, a model normalizing noisy, transliterated text into standard Mongolian faces a difficult challenge when attempting to normalize OOV words. Therefore, we enhanced basic character-level seq2seq models by combining them with various methods to solve the challenges presented by OOV words.

2.3 Data and tools

2.3.1 Training and test data

- *Training data for the character-level seq2seq models*

  Two datasets were prepared for our experiment. The first contained real data consisting of 2,200 sentences written in Mongolian using the Latin alphabet, which were collected from social media sites. These sentences were divided into words, which were then converted into Cyrillic. Canonical versions of the noisy words were then created manually. Our second data set consisted of standardized Latin alphabet transliterations of Cyrillic words which are frequently transliterated in different ways, except that the Latin characters ‘ü’ and ‘ö’ were not used in the standardized transliterations since they are rarely used on social media. Although writers often use incorrectly transliterated text on social media, standard transcriptions of all Cyrillic letters except ‘ү’ and ‘ө’ were used. The first four columns of Table 2.7 show the standard transcriptions of four Cyrillic letters. The two Latin letters ‘u’ and ‘o’ are generally used to represent these four Cyrillic letters, as shown in the two columns on the right of Table 2.7.

Table 2.7. Standard and non-standard transcriptions of four Cyrillic letters

Cyrillic letter	Standard transcription	Cyrillic letter	Standard transcription	Non-standard transcription	Possible Cyrillic originals
у	u	ү	ü	u	‘ө’, ‘у’, ‘ү’
о	o	ө	ö	o	‘о’, ‘ө’

We collected 7,267 Cyrillic words which begin with these four letters, and then transliterated these words correctly into the Latin alphabet using the MNS 5217:2012 standard. In other words, we created standard transliterations of Cyrillic words using the Latin alphabet, thus our second data set did not contain any real or noisy text data. After we created standard transliterations, we changed all occurrences of the Latin letters ‘ü’ and ‘ö’ contained in the standard transliterations into ‘u’ and ‘o,’ respectively.

Our training data consisted of these two datasets, which we collectively called the "word pairs" dataset. The details of this training dataset are shown in Table 2.8. Note that this same training data was used for both the neural and statistical models evaluated in our experiment.

Table 2.8. Details of 'word pairs' training dataset

Description	Noisy data		Clean data		Total data	
	Input	Output	Input	Output	Input	Output
	Noisy Latin transliterations	Cyrillic words	Standard Latin transliterations*	Cyrillic words	Latin words	Cyrillic words
Total words	16,806	17,756	7,267	7,267	24,073	25,023
Total characters	91,172	93,835	60,612	56,879	151,784	150,714
Distinct words	7,548	5,474	7,151	7,267	14,197	12,030
Max. sequence length (words)	17	22	21	19	21	22

* except the letters 'ü' and 'ö' were changed to 'u' and 'o,' respectively.

- *Training data for the word-level language model*

  We then created a target language monolingual corpus of Cyrillic sentences, which was used to train the word-level language model used in most of our proposed methods. The details of this corpus are shown in Table 2.9.

Table 2.9. Monolingual corpus of Cyrillic used to train the language model

Description	
Total sentences	24,000
Total distinct words	27,960
Total words (all words contained in sentences)	225,596

- *Training data for the transliteration model (for SMT model)*

  We then created one additional data corpus for this study. This is a transliteration corpus used to train the transliteration model of our baseline phrase-based SMT method. The details of this corpus are shown in Table 2.10. It contains word alignments consisting of standard Latin alphabet transliterations of Cyrillic words using the MNS 5217:2012 standard, except that the Latin letters 'ü' and 'ö' are omitted.

Table 2.10. Data corpus used to train the transliteration model (for SMT model)

Description	Mongolian words with standard Latin transliterations	Cyrillic words
Total distinct words	7,680	7,680
Total characters	62,882	57,959
Unique tokens	23	34

- *Test data*

  We also prepared test data, which consisted of 200 Mongolian sentences written in noisy Latin characters, which were collected from social media sites. The details of the test data are shown in Table 2.11. In addition to the number of words, characters, and distinct words, the table also contains information on how many in-vocabulary (IV) and OOV words are contained in the test data.

Table 2.11. Details of test data

Description	Noisy words written in Latin alphabet
Total words	1,663
Total characters	9,112
Distinct words	1,178
IV* words (included in the training data)	970 (58%)
OOV words (not included in the training data)	693 (42%)

2.3.2 Tools

Character-level seq2seq models are effective when a small amount of training data is provided, since the character set (or 'vocabulary') is very small in most languages. And since our target data is noisy and transliterated, there is an increased 'rare word' or OOV problem. Therefore, using a character-level seq2seq model alleviates OOV word issue and overcomes the problem better than a word-level seq2seq model. For these reasons, we used character-level seq2seq models without and with attention for normalizing noisy, transliterated, Mongolian text into a canonical format, and tested several variations of these two basic models. All of the models that we built are briefly described in this section.

2.3.2.1 Basic character-level seq2seq models

Seq2seq models contain encoder and decoder recurrent neural networks (RNNs). The encoder processes an input sequence $x = \{x_1, x_2, .., x_n\}$ and compresses the information into context vectors of a fixed length. The decoder is initialized with context vectors to generate the transformed output $y = \{y_1, y_2, .., y_m\}$. In this way, our character level seq2seq model learns to map user transliterations to their canonical forms.

Our first character-level seq2seq model, which does not incorporate an attention mechanism, is based on the architecture proposed by Sutskever et al. (2014), and is shown in Figure 2.1. After reading each symbol, the encoder generates the hidden state at each time step *i*, as shown in Eq. (1):

$$h_i = f_{enc}(h_{i-1}, x_i). \quad (1)$$

The final hidden state produced by the encoder, h_n, is passed on to the decoder and used as the initial hidden state of the decoder ($s_0 = h_n$). The hidden state of the decoder at time step *i* is computed using Eq. (2):

$$s_i = f_{dec}(s_{i-1}, y_{i-1}). \quad (2)$$

Note that both f_{enc} in Eq. (1) and f_{dec} in Eq. (2) are nonlinear functions. The conditional distribution of the next symbol is defined as shown in Eq. (3), where g is a nonlinear function.

$$P(y_i|\{y_1, y_2, \dots, y_{i-1}\}, x) = \text{softmax}\big(g(s_i)\big) \quad (3)$$

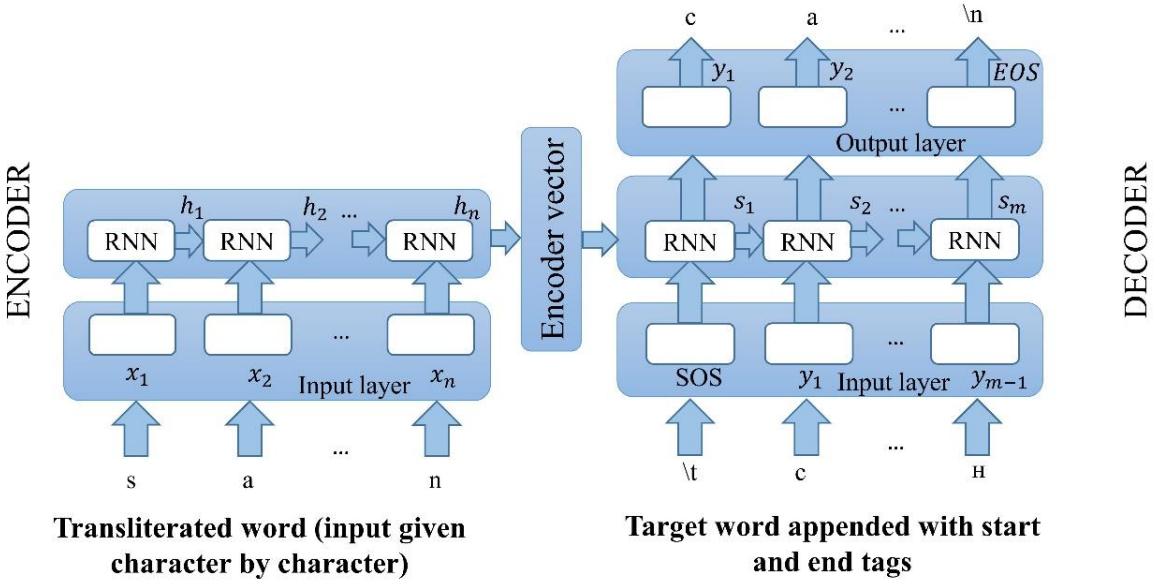

Figure 2.1. Architecture of character-level seq2seq model without attention

Our second character-level seq2seq model, which uses an attention mechanism, is based on the architecture proposed by Bahdanau et al. (2014) and is shown in Figure 2.2. The encoder generates the hidden state of each element in the input sequence $h_i = f_{enc}(h_{i-1}, x_i)$, $i = 1,2,..,n$. Bahdanau's alignment score function, shown in Eq. (4), is then used to calculate alignment scores between the previous decoder hidden state s_{t-1} and each of the encoder's hidden states h_i, $i = 1,2,..,n$:

$$e_{ti} = v_a \tanh(W_a[s_{t-1}; h_i]), \tag{4}$$

where v_a and W_a are learnable weights in the neural network. Attention weights are calculated for each hidden state of the encoder, and a softmax activation function is applied to the alignment scores to obtain the attention weights, as shown in Eq. (5):

$$\alpha_{ti} = softmax(e_{ti}) = \frac{\exp(e_{ti})}{\sum_{k=1}^{n} \exp(e_{tk})}. \tag{5}$$

Each of the encoder's hidden states is multiplied by the corresponding attention weight, and the results are then summed to produce a context vector, as shown in Eq. (6):

$$c_t = \sum_{i=1}^{n} \alpha_{ti} h_i. \tag{6}$$

This context vector is used to compute the final output of the decoder for each time step *t*, and is concatenated with the previous decoder output and fed into the decoder RNN. Eq. (7) is then used to calculate the hidden state of the decoder at time step *t*. A new output is then generated using Eq. (8), where g is a nonlinear function that outputs the probability of y_t:

$$s_t = f_{dec}(s_{t-1}, c_t, y_{t-1}) \tag{7}$$

$$P(y_t|\{y_1, y_2, \dots, y_{t-1}\}, x) = g(s_t, y_{t-1}, c_t). \tag{8}$$

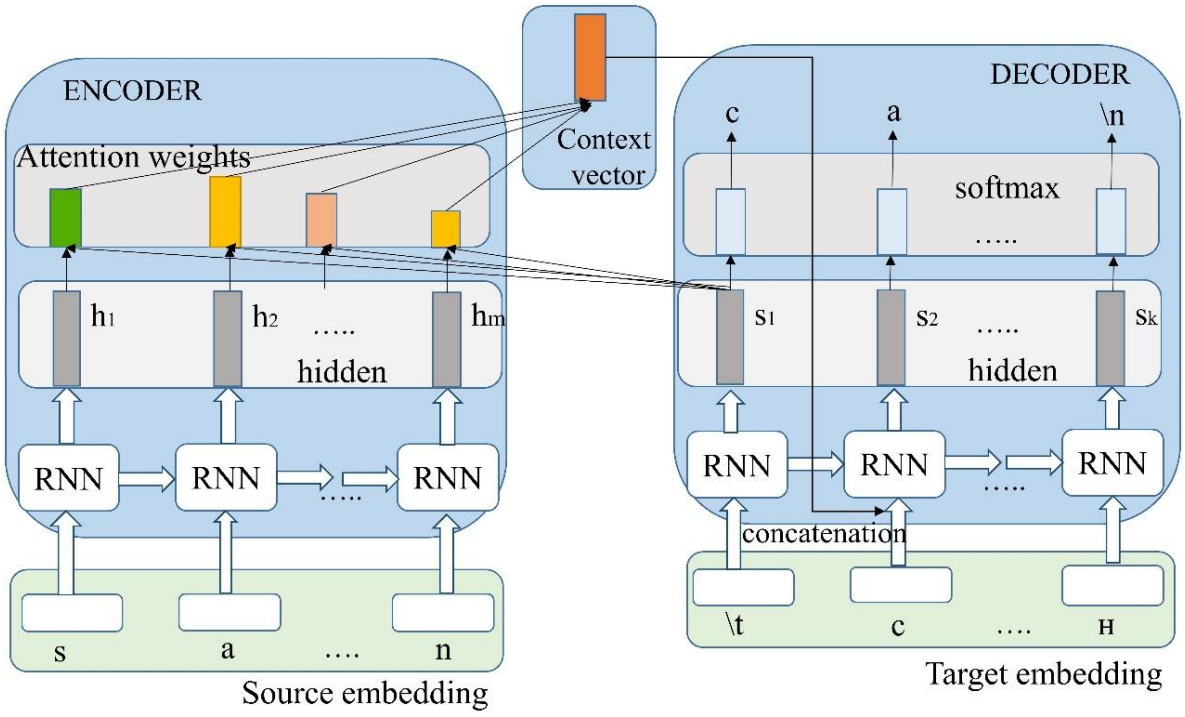

Figure 2.2. Architecture of character-level seq2seq model with attention

In this study, the training dataset described in Table 2.8 in Section 2.3.1 was used to train the basic two character-level seq2seq models and all enhanced character-level seq2seq models.

2.3.2.2 Word-level language model

Most of our proposed methods also use a language model. First, we built an N-gram statistical language model (SLM) which predicts $P(w|h)$, which represents the probability that a particular word (*w*) will occur given a history of previous words (*h*) that contains *N-1* words. The general equation for calculating this probability is:

$$P(w_n|w_{n-N+1}^{n-1}) = \frac{c(w_{n-N+1}^{n-1}w_n)}{c(w_{n-N+1}^{n-1})}. \quad (9)$$

Unigram, bigram, and trigram language models were all evaluated in our experiments, as well as the neural language model (NLM) we developed, which is shown in Figure 2.3. This neural language model learns the probability of the occurrence of a word based on previous sequences of words which have appeared in the text. The model computes the probability of entire sequences such as $P(w_1, w_2, w_3, \dots, w_n)$ using the following equation:

$$P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2) \dots P(w_n|w_1^{n-1}) = \prod_{k=1}^{n} P\left(w_k \middle| w_1^{k-1}\right). \quad (10)$$

The monolingual corpus described in Table 2.9 in Section 2.3.1 was used to train these language models. Most of our proposed methods predict or generate more than one hypothesis or candidate, and the word-level N-gram language model is then used to select the most suitable hypothesis or candidate based on context.

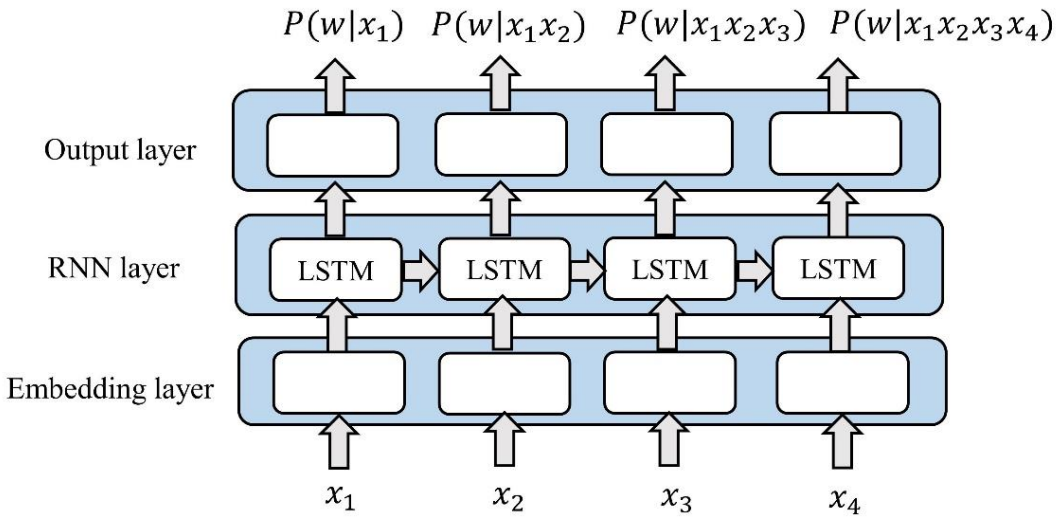

Figure 2.3. Architecture of neural language model

2.3.2.3 Baseline method: SMT

In this section we provide a short description of our baseline method, which is based on a conventional phrase-based SMT model, created using the Moses tool developed by Koehn et al. (2007) (http://www.statmt.org/moses/). The SMT model generates translations based on statistical models, which include a translation model and a language model, whose parameters

are derived from the analysis of bilingual text corpora. We used phrase-based SMT for our noisy text normalization task. Our "word pairs" training dataset and monolingual corpus of Cyrillic, described in Tables 2.8 and 2.9 in Section 2.3.1, respectively, were used to train the translation model and language model, respectively. We used the KenLM language model included in the Moses tool. The transliteration module described in Durrani et al. (2014) has been integrated into Moses, and this module is completely unsupervised and language independent. Moses builds the transliteration model (TM) from a transliteration corpus. In this study, the transliteration corpus described in Table 2.10 in Section 2.3.1 was used to train the TM. First, we used the TM to normalize all of the words in the test data. Second, the TM was used to normalize only the OOV words in the test data when using SMT. These two results were compared with the outputs of the neural models. Figure 2.4 shows the architecture of the SMT model.

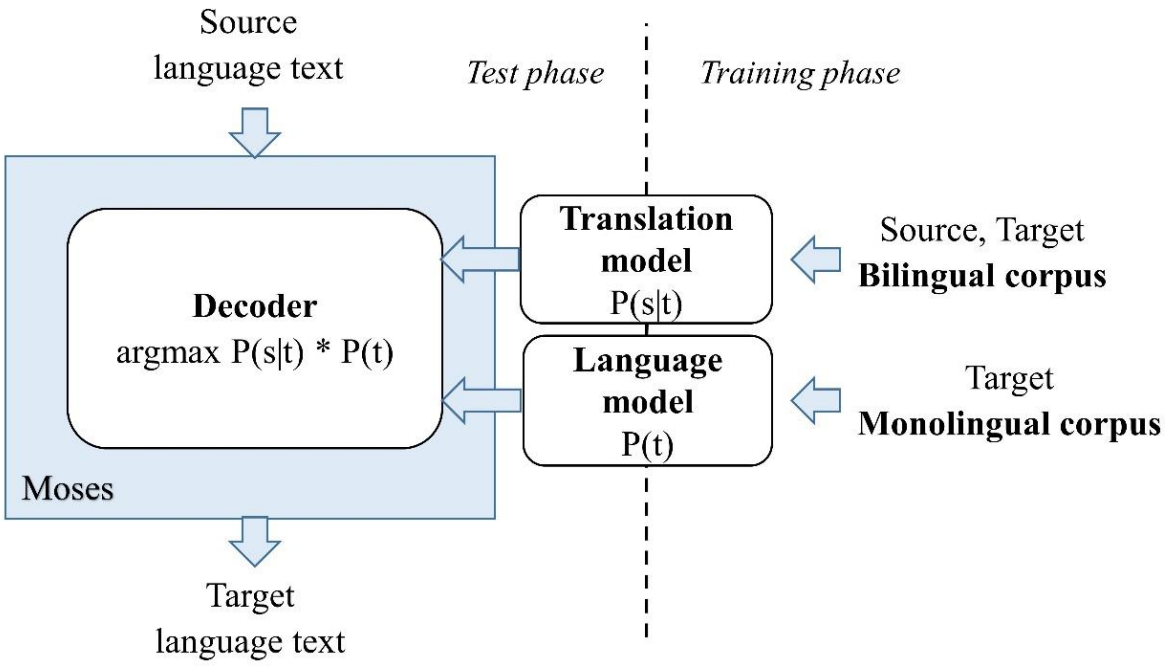

Figure 2.4. Architecture of SMT model

The decoder calculates $P(t|s)$, where *t* is the translation result of *s*. After using Bayes' theorem, the problem can be expressed as follows:

$$P(t|s) \propto P(s|t)P(t). \tag{11}$$

Translation model $P(s|t)$ is the probability of a particular translation being accurate, and language model $P(t)$ is the expression of the fluency of the sentence. The system outputs the best translation $\tilde{t}$ by picking the translation with the highest probability, as shown in Eq. (12), where *T* is the set of all target strings:

$$\tilde{t} = \arg\max_{t \in T} P(t|s) = \arg\max_{t \in T} P(s|t)P(t). \quad (12)$$

2.4 Methodology

2.4.1 Introduction

This chapter explains a more detailed description of how each method operates. The contribution of this study is the exploration of variants of character-level seq2seq models, in search of methods that can outperform conventional methods when using a small amount of training data to normalize noisy, transliterated text, while accurately handling OOV words. In this study, we compare the performance of several of these models with each other and with conventional baseline methods. The first two of our neural models are the basic seq2seq models described in Section 2.3.2.1. But since both of these models use a greedy search method, they are not robust when used on noisy, transliterated text, which is more difficult to normalize than noisy text in its native alphabet. For example, there are several different Cyrillic letters which can be substituted for particular Latin letters due to the chaotic use of the Latin alphabet in noisy, transliterated, Mongolian text, and choosing only one best hypothesis can lead to poor results. The use of Latin alphabet alternatives instead of Cyrillic words also increases the number of OOV words. Therefore, canonical forms of OOV words can actually be IV words, and OOV words can be converted into more than one IV word. Another challenge is working with a limited amount of training data. For these reasons, it is difficult to normalize OOV words effectively using the basic seq2seq models, making the question of how best to normalize OOV words when we have limited data the most important issue. Because of the difficulties described above, we tried combining seq2seq conversion with different beam search strategies, N-gram based context adoption, edit distance-based correction and dictionary-based checking in novel ways, in order to improve performance.

2.4.2 Selection methods

Many NLP applications such as neural machine translation, text summarization and caption generation involve generating sequences of words. Beam search algorithm is widely used to improve output of these text generation tasks. Most of our proposed methods used the beam search algorithm, which selects multiple hypotheses instead of only the one best hypothesis for an input sequence at each time step. This is more appropriate for our task, where we are normalizing noisy text that is also transliterated. Important issues include determining when the beam search will be finished, as well as which hypothesis should be selected. Bahdanau et al. (2014) used a "shrinking beam" method. In this method, when a completed hypothesis is found, beam size shrinks by one. If beam size shrinks to zero or the number of steps hits a hard limit, beam search would finish. The completed hypothesis with the best score in all completed hypotheses is returned. Klein et al. (2017) used a different strategy of beam search. In their way, beam search terminates whenever the highest-ranking hypothesis in the current step is completed, without considering any other completed hypotheses. Thus, the current highest-ranking hypothesis is returned.

In our case, the beam search is finished when the loop reaches the maximum length of the output when using the inference time checking method. However, when using our other methods, the beam search is repeated until the first *N* completed hypotheses are obtained. After finishing the beam search, we select the final output from among the hypotheses or candidates using one of the following methods:

- **Selection Method 1:** Before selecting the final output, the first *N* completed hypotheses are checked for canonical accuracy using the dictionary that was created with a monolingual corpus, which is described in Table 2.9 in Section 2.3.1. If all of the completed hypotheses are considered incorrect and not canonical, the first completed hypothesis is the final output of the seq2seq model. If there are canonical hypotheses, the hypothesis with the highest score among all of the canonical, completed hypotheses is the final output of the seq2seq model.
- **Selection Method 2:** Noisy, transliterated words can sometimes be normalized into more than one canonical word. For example, the word 'xur' can be normalized into

three different canonical words: ‘xyp,’ ‘xөp,’ and ‘xγp.’ In addition, the edit distance method generates candidates which are canonical words from incorrectly normalized words. Therefore, in order to select the correct word, we need context information, so a word-level N-gram language model is used to select the most suitable hypothesis or candidate based on context. The language model calculates the probability of the appearance of each given sequence, consisting of a hypothesis or candidate and the two previous words. The hypothesis or candidate with the maximum probability is the final output.

- **Selection Method 3:** When finishing the beam search, the hypothesis with the highest score is the final output of the seq2seq model.

2.4.3 Description of each method

We used a combination of greedy and beam searches to improve the output of the model. If the one best hypothesis obtained using a greedy search was considered incorrect, multiple hypotheses are then selected using a beam search. This hybrid method can be more effective than using only a beam search. In other words, since the seq2seq model can usually normalize IV words correctly, multiple hypotheses are only selected when the noisy, transliterated text is normalized incorrectly, possibly because it is an OOV word, optimizing the operation of the model.

Also, if the text is not transliterated, text normalization might be similar to the task of spell correction, which is even more difficult to perform with noisy text, even when it is not transliterated. Although the results of the basic seq2seq models are not useful for normalizing noisy, transliterated text, the model might be able to predict output close to the target from the input. Therefore, we applied spell correction techniques to the results of the basic seq2seq models to detect and correct incorrectly normalized words, improving the normalization of OOV words.

Our best performing method can be considered a dictionary-based method, as it optimizes the prediction process by checking each hypothesis for spelling errors at each time step using a dictionary, thus reducing the number of incorrect hypotheses and retaining only the correct

ones for the next step. While our other methods check, correct, and select hypotheses after they have already been predicted, the unique feature of this method is that the model checks and selects hypotheses every time a character is generated. In other words, the output of the model can be improved every time the next symbol is created, as well as when finishing the beam search, the hypothesis with the highest score will be optimal output. As a result, this method is more efficient than the others.

Each of our proposed methods is described below in more detail:

- **Simple seq2seq model:** First, we built our basic character-level seq2seq models without and with attention, and did not use any additional methods with these two basic models. We designated these models **M1 (w/o attention)** and **M2 (w/ attention)**.
- **Seq2seq with beam search:** The next two methods employ the character-level seq2seq models described above with the addition of a beam search. These models were designated **M1 with BS** and **M2 with BS**. The seq2seq model computes a score for the next appearance of each symbol y_t ($y_t \in V$), based on the previous sequence $\{(y_1, y_2, \dots, y_{t-1})^i\}, \mathrm{i} = 1, 2, .., N$. *V* is a vocabulary that contains all of the characters in the training data, and *N* is equal to the beam size. At time step *t*, the scores of all hypotheses $\{(y_1, y_2, \dots, y_{t-1})^i y_t\}$, $\mathrm{i} = 1, 2, .., N$, $(y_t \in V)$ are calculated by summing up the scores of the previous sequence $\{(y_1, y_2, \dots, y_{t-1})^i\}$, $\mathrm{i} = 1, 2, .., N$ and the score of the next symbol y_t. Then, all of the hypotheses are sorted according to their respective scores from the character-level seq2seq model. The top *N* hypotheses are then selected as inputs for the next step in each model. The beam search process is repeated until the first *N* completed hypotheses are obtained. After finishing the beam search, we have *N* hypotheses $\{(y_1, y_2, \dots, y_{m_i})^i\}$, $\mathrm{i} = 1, 2, .., N; m_i \leq D$, where *D* is the maximum length of an output sequence. **Selection Method 1** is then used to select the optimal hypothesis from among the *N* completed hypotheses.
- **Seq2seq with modified search:** Next, we slightly modified the previous two methods (**M1 with BS** and **M2 with BS),** so that both a greedy search and a beam search are used. These two methods were designated **M1 with GS and BS** and **M2 with GS and BS**. The beam search is only used when the output of the greedy search is considered

incorrect and not canonical. The output of the greedy search is checked for canonical accuracy using a dictionary, and if the output is canonical, it is the final output of the seq2seq model. If the results of the greedy search are rejected, both of these modified search methods then operate in the same manner as the previous two methods.

- **Use of NLM score:** These methods combine the use of a NLM with the previously described methods which use either a beam search or a combination of a greedy search and beam search. All of the processes for these models are the same as those used in method **M1/M2 with BS**, except that after finishing the beam search, **Selection Method 2** is used to select one of the first *N* completed hypotheses. In other words, the NLM is used to select the best final output of the character-level seq2seq model. Note that the seq2seq models and NLMs were trained separately. [17] integrated the scores of the word-level NLM and the character-level neural model. In our method however, the scores of these models are used separately in each method. In Table 2.12 in Section 2.5.2.1, these methods are designated by the same names as the previously described methods, but we note that NLM was applied, indicating that only the score of the NLM was used in the final ranking. On the other hand, the previously described methods are categorized under the designation 'w/o NLM.' It would be a natural extension to integrate the scores of both the character-level seq2seq model and the word-level NLM in order to select one of the first *N* completed hypotheses. Thus, we rescored these *N* hypotheses with the weighted sum of the scores calculated using the character-level seq2seq models and the NLM. However, this score integration approach did not work well, and we obtained the same results as when using our previously described methods (M1/M2 with beam search and M1/M2 with modified search). Therefore, we do not show the results of this score integration approach in Table 2.12 in Section 2.5.2.1.
- **Using edit distance and language model:** When using SMT for normalizing noisy text, the TM learns the mapping between two sets of characters, which is quite useful for normalizing OOV words. We used the post-decoding transliteration method of SMT (Method 2 in Durrani et al. (2014)), which generates the list of OOV words automatically by running the decoder, while SMT only normalizes the IV words. After

outputting the OOV word list, SMT normalizes these OOV words using the TM, and then selects the best transliteration from a list of *N*-best transliterations using transliteration and language model features in a post-decoding step. When using these two methods, i.e., edit distance and a language model, we tried to correct words which were incorrectly normalized by the character-level seq2seq model in the same manner that SMT normalizes OOV words. Thus, our methods which we designated as **M1+ ED and LM (SLM or NLM)** and **M2 + ED and LM (SLM or NLM)** have two stages. The basic character-level seq2seq model is used to normalize noisy, transliterated text in the first stage. We then try to correct incorrectly normalized words in the second stage using edit distance and a word-level SLM or NLM as in Norvig (2016), which correspond to the transliteration and language models used in SMT, respectively. In other words, the second stage attempts to improve the output of the first model. We applied one and two edits in order to generate all of the possible candidates from each incorrectly normalized word. The dictionary was used for incorrectly normalized word detection and candidate generation. We used the same method for detecting incorrectly normalized words and generating possible candidates as Saloot et al. (2014) used to detect OOV words and generate candidates. However, we did not follow their approach regarding candidate selection using a language model. In addition, we used a neural model in the first stage, while Saloot et al. (2014) did not use a neural model for noisy text normalization. Our **Selection Method 2** was then used to select the most appropriate candidate from all of the possible candidates.

- **Inference time checking:** This performance enhancement method involves checking each hypothesis found using a beam search at each time step during the inference process. These methods are designated as **M1 (inference time checking)** and **M2 (inference time checking)**. In these methods, at time step *t* the scores of all hypotheses $\{(y_1, y_2, \dots, y_{t-1})^i y_t\}$, $\mathrm{i} = 1, 2, .., N, (y_t \in V)$ are calculated by a sum of the scores of the previous sequence $\{(y_1, y_2, \dots, y_{t-1})^i\}$, $\mathrm{i} = 1, 2, .., N$ and the score of the next symbol y_t. After every symbol y_t, $(y_t \in V)$ is predicted at time step *t* from the previous sequence $\{(y_1, y_2, \dots, y_{t-1})^i\}$, $\mathrm{i} = 1, 2, .., N$, and the incomplete predicted sequence $\{(y_1, y_2, \dots, y_{t-1})^i y_t\}$ $(y_t \in V)$ is then checked using the dictionary to determine if it is

correct or not. The sequence is matched with the prefix of each word in the dictionary, and if the sequence is determined to be incorrect it is pruned. The objective is to retain only correct sequences for the next step. After all of the hypotheses are sorted according to their respective scores, the top *N* hypotheses are selected as the inputs for the next step. The beam search process ends when the loop reaches the maximum length of the output, which is 22 in this case. Our **Selection Method 3** is then used to select the hypothesis. This method, **character-level seq2seq with attention (M2)**, achieved the best normalization results of the methods evaluated in this study.

2.5 Implementation

2.5.1 Parameter settings

The character-level seq2seq model without attention uses several hyperparameters, which can be described as follows. In both the encoder and decoder models, we used three layers of stacked GRUs containing 256 hidden units each, with a dropout of 0.5. An RMSprop optimizer with a learning rate 0.001 was also used, with the batch size set to 32. We created the character-level seq2seq model using Keras (Chollet et al. 2015).

The character-level seq2seq model with attention was built using TensorFlow (Abadi et al. 2016), and both the encoder and decoder models use a single GRU layer with 512 hidden units, while the size of the embedding vector is 250 dimensions. An Adam optimizer with a learning rate of 0.0002 and a batch size of 32 was also used.

In the word-level NLM, an Adam optimizer with a learning rate of 0.001 was used. The model had a single LSTM layer with 100 hidden units, and dropout was 0.5. The size of the embedding vector was 50 dimensions, and the batch size was 32. We also used the Keras library to build this model.

2.5.2 Results and analysis

2.5.2.1 Result-1

We conducted normalization experiments and compared the performance of our various character level neural seq2seq models with the performance of conventional baseline methods. Table 2.12 shows the results of these experiments. The first two models, the TM only of the SMT, and a conventional SMT, were our baselines. The next two models (M1 and M2) are the basic versions of our seq2seq model, without and with attention, respectively. The next two models are the two versions of our basic seq2seq model (without and with attention) using a beam search. The following two seq2seq models are the same two versions of our basic model using both greedy and beam searches. In the four models just described (models 5, 6, 7 and 8), the NLM score was not used. But in the following four seq2seq models in Table 2.12 (models 9, 10, 11 and 12), the scores of the seq2seq model and the NLM were used separately. In the next four seq2seq models (models 13, 14, 15 and 16), we used our two basic models with edit distance and either an SLM or NLM in the second stage. The last two models (models 17 and 18) used our two basic models plus inference time checking, to check all of the incomplete hypotheses during the inference period in order to determine whether or not they were correct, in an attempt to retain only the correct hypotheses. All of the neural models were trained using the same training corpus.

We used word error rate (WER) and character error rate (CER), as defined in Eqs. (14) and (15), respectively, which are metrics derived from the Levenshtein distance (Levenshtein 1966), to evaluate the performance of each model. The total number of insertions, deletions, and substitutions that were needed to transform the normalized string into the reference string is used to represent the total number of incorrect words or characters, respectively, as defined in Eq. (13). WER and CER are calculated by dividing the number of incorrect words or characters, respectively, by the total number of words or characters in the reference string, as follows:

$$Number\ of\ incorrect\ words\ or\ characters\ =\ I\ +\ D\ +\ S \tag{13}$$

I = Number of **I**nsertions D = Number of **D**eletion S = Number of **S**ubstitution

$$WER = \frac{Number\ of\ incorrect\ words}{Total\ words} \quad (14)$$

$$CER = \frac{Number\ of\ incorrect\ characters}{Total\ characters} \quad (15)$$

Table 2.12. Comparison of WERs and CERs for each method when using test data

Model		WER	CER	Model #
Baseline model: TM of SMT		19.49%	8.07%	1
Baseline model: SMT		17.92%	7.40%	2
M1 (w/o attention)		22.49%	7.65%	3
M2 (w/attention)		22.13%	6.97%	4
w/o NLM	M1 with BS	17.08%	7.68%	5
	M2 with BS	16.90%	9.05%	6
	M1 with GS and BS	16.72%	6.89%	7
	M2 with GS and BS	15.70%	7.73%	8
w/ NLM	M1 with BS	20.87%	7.85%	9
	M2 with BS	21.11%	8.40%	10
	M1 with GS and BS	17.20%	6.96%	11
	M2 with GS and BS	16.06%	7.19%	12
M1 + ED and SLM		16.66%	7.87%	13
M2 + ED and SLM		15.28%	6.77%	14
M1 + ED and NLM		14.92%	7.12%	15
M2 + ED and NLM		13.96%	**6.26%**	16
M1 with inference time checking		14.38%	7.19%	17
M2 with inference time checking		**13.41%**	6.56%	18

- TM – Transliteration model
- SMT – Statistical machine translation
- M1 – Seq2seq model without attention
- M2 – Seq2seq model with attention
- GS – Greedy search
- BS – Beam search
- NLM – Neural language model
- SLM – Statistical language model
- ED – Edit distance
- Underlined scores = achieved lower WERs than the baseline methods
- Bold scores = best performance in that category

When only the TM of the SMT was used to normalize noisy text, the results were worse (i.e., the error rates were higher) than when using the entire SMT model. Our two basic seq2seq models (without and with attention) achieved the worst word-level performance, with higher WERs than all of the other models. The two versions of our basic seq2seq model using a beam search followed by an NLM (methods 9 and 10) achieved word-level results slightly better than the two basic seq2seq models, but worse than the two baseline models. All of our other proposed seq2seq models achieved higher word-level performance than the baseline models. The two-stage methods (models 13, 14, 15 and 16 in Table 2.12) achieved word-level performance higher than the baseline methods and higher than some of the other neural methods. The inference time checking method (model 17), which checked each hypothesis during the inference period, achieved word-level performance higher than the baseline methods and all of the neural methods except for models 16 and 18. The other inference time checking method, using the seq2seq model with attention (model 18), achieved higher word-level performance than the baseline methods and also outperformed all of the other neural models.

In all of the experiments, the seq2seq models with attention (except for M2 with BS followed by the NLM) achieved better word-level performance than the seq2seq models without attention. Regarding models 5 through 12 in Table 2.12, the word-level results for models followed by the NLM were lower than the models without the NLM. In addition, the

seq2seq models which used greedy search and beam search (models 7, 8, 11 and 12), achieved better word-level performance than the models which used only a beam search (models 5, 6, 9 and 10), regardless of whether or not the NLM score was used.

The two-stage methods (models 13, 14, 15 and 16 in Table 2.12) used one of the basic character-level seq2seq models in the first stage. We can see that the word-level performance of the basic seq2seq models were lower than the baselines, but that character-level results were nearly the same or slightly higher than the baselines. This was because applying the spelling corrector to the output of the first stage improved word-level performance, which means that the basic seq2seq model predicted outputs close to the targets, and that errors involving incorrectly normalized words were then easily corrected using a spelling corrector. The two-stage methods were competitive with the inference time checking methods (models 17 and 18), however the inference time checking methods took longer than the other methods when normalizing noisy, transliterated text.

A total of 12 models, whose results are underlined in Table 2.12, achieved higher word-level performance than the baselines. The best WER (13.41%) and CER (6.26%) were obtained using different methods (inference time checking and edit distance, respectively).

2.5.2.2 Result-2

Next, we analyzed how the different methods normalized IV and OOV words in the test data. As shown in Table 2.11 in Section 2.3.1, our test data contained 1,663 words, which included 970 IV words which appeared in the training data and 693 OOV words which did not, i.e., 58% of the test data were IV words and the remaining 42% were OOV words. Table 2.13 shows how well each of the evaluated methods normalized IV and OOV words. We can see that the differences in the WERs for IV words among the neural models were small, while the differences in the WERs for OOV words were large. Results for IV words for most of the neural models are close to the results for the basic seq2seq models, but with slight improvement. However, the minimum to maximum WERs for normalization of OOV words for all of the neural models ranged from 26.12% to 47.05%, with the basic seq2seq models achieving the lowest performance. This indicates that if the amount of training data is small, and the rules for writing noisy, transliterated text are not limited, the basic seq2seq models face a difficult

challenge when attempting to normalize OOV words. The OOV word-level normalization performance of all of the other proposed neural models were higher than the basic seq2seq models however, and most of the neural models outperformed the baseline models. Therefore, our experimental results show that the proposed performance enhancement methods we tested improved the robustness of the existing neural models for normalizing OOV words in low resource scenarios. SMT achieved the best performance when normalizing IV words, however our proposed inference time checking method achieved the best performance when normalizing OOV words.

Table 2.13. Comparison of WERs and CERs for IV and OOV words for each method when using test data

Model description		WER		CER		Model #
		IV words	OOV words	IV words	OOV words	
Baseline model: TM in SMT		6.40%	37.81%	2.76%	12.81%	1
Baseline model: SMT		3.72%	37.81%	1.35%	12.81%	2
M1 (w/o attention)		4.95%	47.05%	1.81%	12.86%	3
M2 (w/attention)		5.06%	46.04%	1.90%	11.49%	4
w/o NLM	M1 with BS	4.13%	35.21%	1.46%	13.22%	5
	M2 with BS	4.75%	33.92%	1.90%	15.43%	6
	M1 with GS and BS	4.03%	34.49%	1.44%	11.76%	7
	M2 with GS and BS	4.33%	31.61%	1.57%	13.22%	8
w/ NLM	M1 with BS	6.71%	40.70%	2.25%	12.84%	9
	M2 with BS	7.74%	39.83%	2.52%	13.65%	10
	M1 with GS and BS	4.13%	35.50%	1.46%	11.86%	11
	M2 with GS and BS	4.13%	32.76%	1.48%	12.27%	12
M1 + ED and SLM		4.44%	33.77%	1.79%	13.30%	13
M2 + ED and SLM		4.85%	29.88%	1.90%	11.11%	14
M1 + ED and NLM		4.44%	29.59%	1.79%	11.88%	15
M2 + ED and NLM		4.75%	26.84%	1.86%	**10.19%**	16
M1 with inference time checking		3.92%	29.01%	1.39%	12.35%	17
M2 with inference time checking		4.33%	**26.12%**	1.57%	11.01%	18

2.6 Conclusion

In this study we investigated the performance of variants of neural models when normalizing noisy, transliterated Mongolian text in a low resource scenario, and compared the performance of our proposed methods with that of two conventional methods, a TM and an SMT method. Twelve of our neural models achieved word or character-level normalization performance that was better than the baseline models. Specifically, all of the methods, except for the two basic seq2seq models (without and with attention) and the two seq2seq models with beam search followed by an NLM, achieved better word-level normalization performance than the conventional methods. The best normalization performance was achieved by our inference time checking method, which included the checking of each hypothesis during the inference period. It achieved a WER of 13.41% when using the test data, outperforming the baseline SMT model by 4.51%. Most of the methods we proposed improved robustness when normalizing OOV words, and all of them achieved higher word-level performance than the basic seq2seq models. Although the baseline methods achieved better word-level performance when normalizing IV words, most of our proposed neural methods outperformed the baseline methods when normalizing OOV words, with our inference time checking method achieving a WER 10% lower than the baseline methods when attention was applied.

Chapter 3: Speech synthesis for low-resource language using cross-lingual transfer learning and data augmentation

3.1 Literature review

Recently proposed TTS models based on deep learning techniques (Wang et al. 2017; Shen et al. 2018; Ping et al. 2018; Sotelo et al. 2017) are capable of synthesizing natural, human-like speech. These models require a large amount of speech data for training however, as well as substantial computational power, thus data sparsity is a challenge when developing advanced TTS systems for low-resource languages. Thus, recent studies have proposed a variety of techniques which can be used for TTS with low-resource languages. These techniques include:

- *Monolingual transfer learning*: When there is only a small dataset of a particular type of speech available, such as the speech of an additional speaker, emotional speech data, alternative speaking style data, etc., a pre-trained model, trained using a large amount of a different type of speech data, can be used as a low-resource speech model by using transfer learning. Tits et al. (2019) explored transfer learning for TTS with low-resource, emotional speech. After training their model with a large dataset, they fine-tuned it using a small, neutral speech dataset from a new speaker. They then adapted the resulting model by training it with a small, emotional speech dataset also created using the new speaker. Bollepalli et al. (2019) used the same method as in Tits et al. (2019), except that Lombard speech was used for transfer learning instead of emotional speech. They adapted a pre-trained TTS system using 2 hours of normal speech data from a new speaker. They then adapted the normal speech model for the new speaker to a different speaking style from the same speaker, such as Lombard speech, using a transfer

learning method. In studies Tits et al. (2019) and Bollepalli et al. (2019), all of the datasets used were in the same language.

- *Cross-lingual transfer learning*: Since large amounts of data are often unavailable for low-resource languages, most of the proposed approaches for TTS for these languages have used cross-lingual transfer learning to train their target language TTS systems. However, when using cross-lingual transfer learning, input space mismatches can occur. Chen et al. (2019) developed TTS systems for low-resource languages and explored cross-lingual symbol mapping to improve the transfer of knowledge learned previously from a high-resource language dataset. Three methods for cross-lingual symbol mapping were evaluated, and two of these methods, which were denoted "Unified" and "Learned", achieved good results. Their proposed method "Learned" automatically mapped the relationship between source and target language linguistic symbols to transfer knowledge learned previously. To do this, they pre-trained an automatic speech recognition (ASR) system using the source language, then fixed the parameters of the pre-trained ASR system and concatenated their proposed Phonetic Transformation Network (PTN). They used the target language data in this stage, and PTN learned to find the possible target symbols given the ASR output, source symbols. Their results when using their proposed "Learned" method were no better than when using the "Unified" method, but were comparable. In this study, we used two high-resource languages, English and Japanese, and these datasets were used both sequentially and simultaneously when training the model. Therefore, in our approach, we used the "Unified" method. In other words, we converted the transcriptions of all of the utterances in each dataset into their phonetic transcriptions based on IPA, and we then created a unified symbol set to solve the input space mismatch problem.
- *Multi-speaker models*: In addition, multi-speaker models have been used to reduce the amount of training data needed for TTS. Latorre et al. (2019) has shown that multi-speaker models, which use a small amount of data from each speaker, are more effective than speaker-dependent models trained with more data. In Luong et al. (2019), researchers investigated the effect on TTS performance of training a multi-speaker model using a speaker-imbalanced corpus. They found that simply combining all the

available data from every speaker when training the multi-speaker model produced better results than using a speaker-dependent model. Gutkin et al. (2016) constructed a multi-speaker, acoustic database using crowdsourcing, and then used it to bootstrap a statistical, parametric speech synthesis system. These studies all used multi-speaker datasets which were in the same language as the target speech.

- *Multilingual models:* Since high-quality, multi-speaker data is generally unavailable in most low-resource languages, multilingual or multilingual/multi-speaker models can be used to address data availability issues. Yu et al. (2016) proposed a multilingual bi-directional long short-term memory (BLSTM) based speech synthesis method which transforms the input linguistic features into acoustic features. The input layer and hidden layers of the BLSTM were shared across different languages for speech synthesis of low-resource languages, but the output layer was not shared. The input feature vectors of different languages were combined to form a single, uniform representation of the input features. The shared hidden layers transform the uniform input features into an internal representation that can benefit low-resource TTS. Their proposed multilingual BLSTM based speech synthesis method was able to more accurately predict acoustic features than a monolingual BLSTM. Li and Zen (2016) built a long short-term memory (LSTM) recurrent neural network based, multi-language/multi-speaker (MLMS) statistical parametric speech synthesis system using six languages. Their proposed MLMS model achieved similar performance to that of conventional language-dependent and speaker-dependent models. They also demonstrated that adapting their proposed system to new languages using limited training data achieved better performance than building low-resource language models from scratch. Korte et al. (2020) conducted experiments to compare the naturalness of speech from single-speaker models with speech from multilingual models when different amounts of the target speaker's data were used for training. They also compared the naturalness of speech from monolingual, multi-speaker models with speech from multilingual, multi-speaker models when larger amounts of non-target language training data were used. As a result, they demonstrated the effectiveness of using multilingual models to improve the naturalness of speech in low-resource

language TTS systems, finding that the use of foreign language training data improved the quality of low-resource target language speech output. Their proposed multilingual model used a separate encoder for each language to represent language information. They found that this method of representing language information was more effective than using language embedding. Lee et al. (2020) built bilingual, multi-speaker TTS models using two monolingual datasets to investigate how speech synthesis networks learn pronunciation from datasets of different languages. They noticed that two, learned phoneme embeddings were located close together when they had similar pronunciations. Therefore, based on this observation, they proposed a training framework to utilize phonetic information from a different language. They showed that pre-training a speech synthesis model using datasets from both high- and low-resource languages could enhance the performance of the TTS model with low-resource languages. Chen et al. (2019) built a cross-lingual, bilingual TTS system with learned speaker embedding, using two monolingual, multi-speaker datasets. A speaker encoder model, trained with the English and Chinese datasets, was used to represent the latent structure the utterances of different speakers and language pronunciations. The learned speaker embedding extracted by the speaker encoder was then used to condition the spectrogram prediction network. They noted that the learned speaker embedding could represent the relationship between pronunciations across the two languages, even though English and Chinese have different phoneme sets. They observed that phonemes with similar pronunciations were inclined to remain closer to each other across the two languages than to the other phonemes.

- *Data augmentation:* Data augmentation is widely used in ASR to produce additional synthetic training data (Ko et al. 2015; Zhou et al. 2017; Geng et al. 2020). Recent speech synthesis studies have shown that data augmentation can also improve the performance of TTS models. Huybrechts et al. (2021) built high-quality TTS models for expressive speech, to be used when only a very small amount of expressive speech data is available for a target speaker. First, they generated synthetic speech data from a source speaker to the target speaker in the desired speaking style using a voice conversion model. Second, they trained the TTS model using the generated synthetic

speech data and the target speaker recordings. Then the pre-trained model was fine-tuned with non-synthetic data in order to focus on the actual target space more closely. Using both data augmentation and fine-tuning methods improved the signal quality, naturalness, and style adequacy of the synthetic speech without any drop in speaker similarity. Hwang et al. (2021) proposed a TTS-driven data augmentation method to improve the quality of the output of a non-autoregressive (NAR) TTS system. First, they trained the source autoregressive (AR) TTS model using recorded speech data from a professional speaker. Then, text scripts were prepared for generating synthetic data using the source AR TTS model. After generating a large amount of synthetic data (179 hours), this augmented corpus was used to train the target NAR TTS model. The proposed data augmentation method was effective and significantly improved the quality of the output of the NAR TTS system. Cooper et al. (2020) investigated two speaker augmentation scenarios for a multi-speaker TTS model. The first speaker augmentation method creates "artificial" speakers by changing the speed of the original speech using a sound exchange audio manipulation tool (SoX)[1]. The second method uses low-quality data containing background noise and reverberation, which was collected for purposes other than TTS, such as ASR. This low-quality data consisted of four new ASR corpora which included speech in different dialects. They modified the postnet and encoder of the Tacotron model to support the additional channel and dialect factors. The channel represents a factor in the low-quality data jointly caused by the frequency characteristics of the recording equipment, noise and reverberation. Their modified Tacotron model trained with speaker augmentation data improved the naturalness of the synthesized speech of speakers seen during training. They observed that artificial speaker augmentation contributed to speech naturalness rather than speaker similarity, and that adding low-quality data improved the quality of the synthetic speech for seen speakers. Liu et al. (2020) built a bilingual TTS model for use when the amount of target language data was limited. They tried to solve the problems of accent carry-over and mispronunciation. Accent carry-over can occur during cross-

[1] http://sox.sourceforge.net

lingual speech synthesis, so tone preservation mechanisms were used to address this. Mispronunciation during low-resource synthesis occurs when the synthesizer does not have enough examples to learn proper phonetization. They addressed this problem with data augmentation, using noise and speed perturbations to increase the target low-resource language dataset 10-fold. SoX was used for speed perturbation. Their experimental results demonstrated the significant potential of data augmentation for improving speech quality when working with extremely low-resource languages. Our proposed method uses a data augmentation method similar to that used in Cooper et al. (2020) and Liu et al. (2020), but in these studies either two or four additional versions of the original utterances, respectively, were generated by changing the speed factor. Vehicle noise was then added to all of the utterances in Liu et al. (2020). In this study, we generated synthetic speech with a wider range of variation, creating 26 versions of each utterance from the original speech by changing the pitch and speed. While Cooper et al. (2020) and Liu et al. (2020) increased the amount of training data 3-fold and 10-fold, respectively, using data augmentation we increased the amount of target language training data 27 times the size of the original dataset.

3.2 A brief overview of our methods compared to related work

In previous studies of monolingual and cross-lingual transfer learning which appear in the literature, knowledge learned from a large amount of data was transferred and a pre-trained model was adapted with the specific type of speech data in the same language or speech data in another language. In this study, we are proposing a single-speaker TTS system for use in a low-resource scenario, therefore we also used the same method proposed in previous studies, and trained a monolingual, single-speaker TTS model. But in our approach, two high-resource languages, English and Japanese, were used sequentially for pre-training to improve the transfer of linguistic knowledge. Both of the high-resource language corpora we used are publicly available, and contain speech data from a different, single, female speaker. In contrast, our target Mongolian language dataset contains speech data from a single, male speaker. Therefore, our single-speaker TTS model was trained using a different single-speaker dataset

at each training stage, e.g., during the pre-training and fine-tuning stages. The phonemes of the few Mongolian letters which are not contained in English are contained in Japanese, and vice versa. For example, the phonemes of the Cyrillic letters ‘ө’ and ‘ц’ are not contained in English, while the phonemes of the Cyrillic letters ‘у,’ ‘ө’ and ‘л’ are not contained in Japanese. In addition, the Mongolian language belongs to the Altaic family of languages. It has been suggested that Japanese is linguistically related to Altaic, as there are structural similarities and the pronunciations of the phonemes are very similar. English, on the other hand, is an Indo-European language. Therefore, we first used English, then Japanese, to train the pre-trained model, because it is generally more effective to train models using the less similar data first.

In previous studies involving multi-speaker models and multilingual models which appear in the literature, investigators built multi-speaker models using monolingual data in order to reduce the amount of training data needed, while multilingual models were trained using multilingual or multilingual, multi-speaker datasets. In contrast, our aim is to build a monolingual, single-speaker TTS model which can synthesize the Mongolian speech of the male speaker recorded in the target language dataset. But we believe that a multi-speaker model can be used as a component in its development. In this study, since we do not have multi-speaker data for the targeted low-resource language, we instead trained the multi-speaker model with multilingual data, using the same input representation (one speaker per language, with different speakers for each language), and then fine-tuned it to realize the proposed monolingual, single-speaker TTS model. In other words, we used a multi-speaker model and multilingual data to obtain a monolingual, single-speaker model.

Although we used transfer learning in two different situations (with both single- and multi-speaker models) to address the issue of the limited data, we found that 30 minutes of target language training data was insufficient for cross-lingual training. Therefore, we generated a training dataset which was 27-fold larger using data augmentation in order to solve the limited target language data issue, and this dataset was used to train both the single- and multi-speaker models. Since the augmented data can be considered to be from different speakers, it may therefore be more suitable for training a multi-speaker model.

In this study, we propose a single-speaker TTS system for the low-resource language of Mongolian. The contributions of this study are as follows:

1. We explore the TTS model's performance after cross-lingual transfer learning using high and low-resource language datasets. These datasets were used both sequentially and simultaneously during the training of the spectrogram prediction network.
2. We create a large amount of augmented data by changing some of the characteristics of a very small amount of original target language speech data, such as pitch and speed, and evaluate the TTS model's performance when this augmented data is used for training.
3. We show how the performance of our low-resource language TTS model is enhanced by combining the previous two methods.
4. We investigate how much original target language data is needed when training the proposed TTS model in order to achieve the same results as the baseline model trained with a much larger amount of target language data.
5. We demonstrate how augmented data can also be used to train the neural vocoder, in addition to the spectrogram prediction network.

In the experiments related to contributions 1-3 described above, we tested two types of TTS models; single-speaker (M_S) and multi-speaker (M_M).

3.3 Data and tools

3.3.1 Input representation

We chose English and Japanese as our high-resource source languages, and used Mongolian as the low-resource target language in our experiments. The pronunciation of some phonemes in the three languages are similar, therefore learned phoneme embedding can be shared, improving the performance of our low-resource language TTS model. We created a unified symbol set to solve the input space mismatch between the source and target languages before training. The transcriptions of all of the utterances in the English and Mongolian datasets were converted into their phonetic transcriptions based on IPA. For Japanese, the transcripts of all

of the utterances were first converted into Romaji using an online converter[2], and then converted from their Romaji representations into phonetic transcriptions based on IPA. Table 3.1 shows all of the phonemes used in each dataset. Since some phonemes in these three languages have the same pronunciations, there are overlapping phonemes in the source and target languages. On the other hand, some phonemes exist only in a particular source or target language. The number of phonemes which occurred only in the English language was greater than the number of phonemes that existed only in the target language. In contrast, all of the phonemes of the Japanese language are contained in Mongolian. Only one phoneme in the target language dataset, 'ö,' is not contained in either of the high-resource language datasets, while three phonemes, 'l,' 'ʊ' and 'c,' are contained in the data of one of the high-resource languages. Therefore, to create the unified symbol set, the phonemes 'ö' and 'c' were inserted into the English language dataset by replacing the phonemes that sound the most similar in the English source language dataset. We did not replace many phonemes, and each new phoneme replaced only one occurrence of the English language phonemes. This replacement is necessary when the source and target datasets are used sequentially during cross-lingual transfer learning.

[2] https://nihongodera.com/

Table 3.1. Phonemes used in each dataset.

#	English	Japanese	Mongolian	#	English	Japanese	Mongolian
1	a (aɪ, aʊ)	a	a	21	v	v	v
2	b	b	b	22	w	-	-
3	d	d	d	23	z	z	z
4	e	e	e	24	æ	-	-
5	f	f	f	25	ð	-	-
6	g	g	g	26	ŋ	ŋ	ŋ
7	h	h	h	27	ɑ	-	-
8	i	i	i	28	ɔ	-	-
9	j	j (ja, jo, ju)	j (ja, jo, jʊ)	29	ə	-	-
10	k	k	k	30	ɛ	-	-
11	l	-	**l**	31	ɜ	-	-
12	m	m	m	32	ɪ	-	-
13	n	n	n	33	ʃ	ʃ	ʃ
14	o	o	o	34	ʊ	-	**ʊ**
15	p	p	p	35	ʌ	-	-
16	r	r	r	36	ʒ	-	-
17	s	s	s	37	ʤ	ʤ	ʤ
18	t	t	t	38	ʧ	ʧ	ʧ
19	u	u	u	39	θ	-	-
20	-	-	**ö**	40	-	c	**c**

3.3.2 Dataset

English and Japanese were selected as our high-resource source languages. As our English speech corpus, we used LJSpeech (Ito 2017), a public domain dataset consisting of 13,100 utterances, with a total length of 24 hours. Each audio file is a single-channel, 16-bit PCM WAV with a sampling rate of 22,050 Hz. For our Japanese speech corpus, JUST (Sonobe et al. 2017) was used. It is also a public dataset consisting of 7,696 utterances, with a total length of 10 hours of paired text-speech data. We down-sampled each audio file in the corpus to 22,050 Hz. These source corpora feature the voices of different, single, female speakers.

We prepared a target speech corpus using part of a Mongolian language translation of the Bible, which was manually divided into individual sentences. The entire corpus consisted of 8,183 short audio clips of a single, male speaker, with a total length of 12 hours. Each audio file is a single-channel, 16-bit PCM WAV with a sampling rate of 22,050 Hz. We randomly selected 30 minutes of paired text-speech data, consisting of 307 utterances, to use as the target language dataset in our experiments. There are 35 letters in the Mongolian Cyrillic alphabet. We counted the number of occurrences of each letter in the 30 minutes of target language data before the transcriptions were converted into their phonetic representations, in order to explore how the number of occurrences of a letter affects the learning of its pronunciation. Table 3.2 lists the Mongolian letters contained in the 30-minute and 12-hour target language datasets with corresponding phonetic symbols, as well as the number of occurrences and the distribution of each letter. We sorted the list in descending order by the number of occurrences of the letters in the entire dataset. The letter 'щ' is not contained in the target language dataset because it is never used in the Mongolian language - only Russian loanwords contain this letter. Its pronunciation is identical to 'ш'; Russian loanwords which include the letter 'щ' will sometimes be spelled with 'ш.' In addition, the letters 'к' and 'ф' are also only used in foreign words, and thus appear infrequently. Also, the letter 'п' does not appear in the middle or at the end of a Mongolian word, but sometimes appears at the beginning of a word. Therefore, the four consonants 'к,' 'ф,' 'щ' and 'п' are called "special consonants" in Mongolian, and the number of occurrences of these letters is usually small. Although we randomly selected 30 minutes of target language data for training, the distribution of letters within this target language training data is almost the same as the distribution of letters in the entire 12 hours of the target language dataset. The chart in Figure 3.1 shows the distribution of each letter in both the 30 minutes and 12 hours of target language data.

Table 3.2. Occurrences and distributions of Mongolian letters in the 30-minute and 12-hour target language datasets and IPA phonetic symbols

#	Letter	Phoneme	# Occurrences and distribution				#	Letter	Phoneme	# Occurrences and distribution			
			30 minutes		12 hours					30 minutes		12 hours	
1	а	a	1870	11.00%	54403	11.33%	18	м	m	372	2.19%	10349	2.15%
2	н	n, ŋ	1378	8.11%	41565	8.65%	19	в	v	234	1.38%	8175	1.70%
3	э	e	1625	9.56%	38739	8.07%	20	ч	ʧ	190	1.12%	6744	1.40%
4	г	g	1030	6.06%	28254	5.88%	21	з	z	218	1.28%	6691	1.39%
5	р	r	1060	6.24%	26879	5.60%	22	ы	i	148	0.87%	5109	1.06%
6	и	i	686	4.04%	25673	5.34%	23	ж	ʤ	259	1.52%	4890	1.02%
7	д	d	827	4.86%	23810	4.96%	24	ь	i	176	1.04%	4641	0.97%
8	х	h	856	5.04%	22237	4.63%	25	е	j	132	0.78%	4390	0.91%
9	л	l	742	4.36%	21224	4.42%	26	ш	ʃ	114	0.67%	3516	0.73%
10	ү	u	684	4.02%	19969	4.16%	27	ц	c	72	0.42%	2351	0.49%
11	т	t	647	3.81%	19362	4.03%	28	я	ja	27	0.16%	1765	0.37%
12	о	o	805	4.74%	19019	3.96%	29	ю	jʊ	17	0.10%	843	0.18%
13	й	i	613	3.61%	18724	3.90%	30	ё	jo	36	0.21%	727	0.15%
14	у	ʊ	604	3.55%	17868	3.72%	31	к	k	24	0.14%	591	0.12%
15	ө	ö	532	3.13%	14145	2.94%	32	ф	f	5	0.03%	471	0.10%
16	с	s	495	2.91%	13438	2.80%	33	п	p	1	0.006%	401	0.08%
17	б	b	519	3.05%	13330	2.78%	34	ъ	i	1	0.006%	26	0.005%

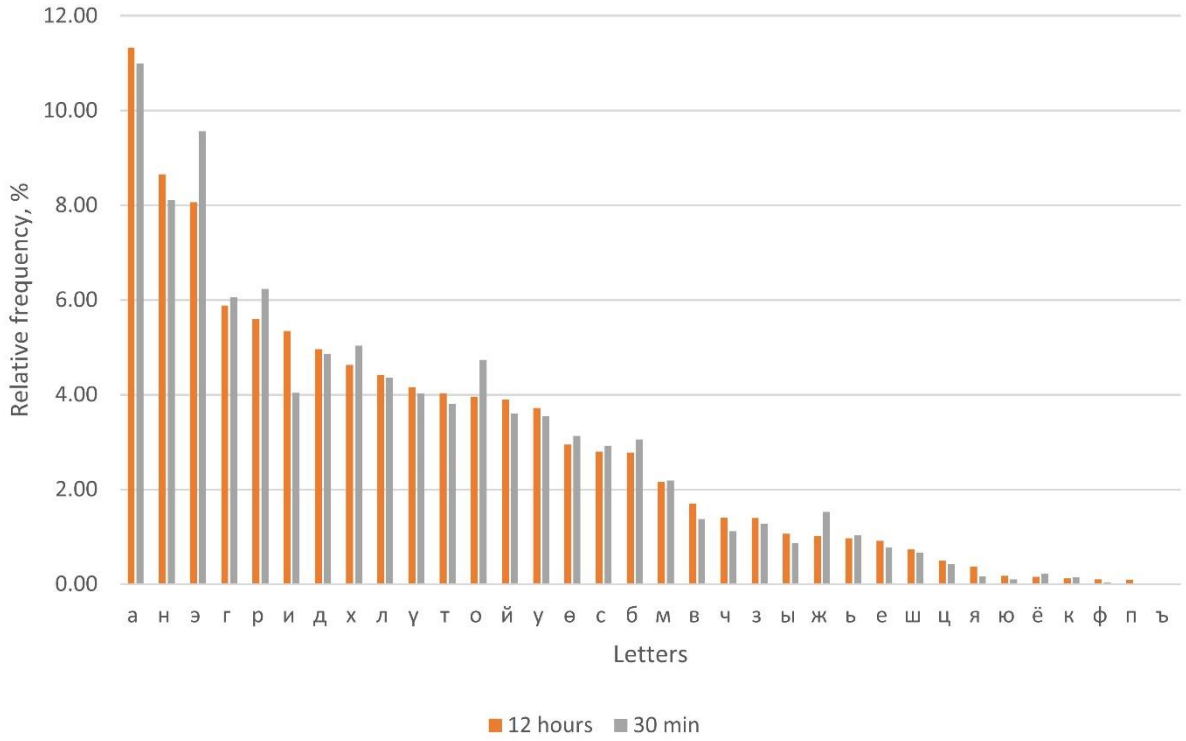

Figure 3.1. Distribution of each letter in 30-minute and 12-hour target language datasets.

We used the entire 12 hours of target language data to train the baseline TTS model (M-MN), which was used for a performance comparison with the proposed models. In addition, the baseline PWG neural vocoder (NV-MN) was also trained using the entire 12 hours of target language data, for a performance comparison with the vocoder trained using augmented speech data (NV-DA).

3.3.3 TTS system

We tested single-speaker and multi-speaker TTS models, trained with high-resource and low-resource language datasets, sequentially or simultaneously, with or without augmented data, to obtain a single-speaker TTS system that is effective when only a limited amount of target language training data is available. Our base TTS system consists of three components: an x-vector speaker encoder, a Tacotron 2-based spectrogram prediction network, and PWG neural vocoder (Yamamoto et al. 2020). We adopted the original Tacotron 2 architecture, which consists of a bi-directional LSTM based encoder and a unidirectional LSTM based decoder

with location sensitive attention, using the same hyperparameters as in Shen et al. (2018), except for the addition of a loss function for guided attention loss (Tachibana et al. 2018), which supports faster convergence. Although a reduction factor (r), representing the number of frames to generate at each decoding step, was not used in Shen et al. (2018), we used the reduction factor r = 1 for the single-speaker model, while the reduction factor r = 2 was used for the multi-speaker model to speed up the training process. Table 3.3 shows the hyper-parameters used in all models. We used a batch size of 32 for all of the models, except the pre-trained, multi-speaker models trained with both high-resource language datasets. The spectrogram prediction network was constructed using the open-source speech processing toolkit ESPnet (Hayashi et al. 2020). We used a pre-trained x-vector (Snyder et al. 2018) for speaker embedding, as provided by Kaldi. The pre-trained speaker encoder was trained on the LibriTTS corpus (Zen et al. 2019). The speaker embeddings were concatenated with each encoder state. PWG is a non-autoregressive neural vocoder trained to minimize multi-resolution, short-time Fourier transform (STFT) loss and waveform domain adversarial loss. We used the public implementation[3] to train the PWG neural vocoder with augmented data created using a very small target language dataset and it was used to generate the waveform in all of our experiments. Figure 3.2 shows an overview of our base TTS system. The speaker embedding network in Figure 3.2 is used to train a multi-speaker model. For the single-speaker model, we used the same network without speaker embedding. Although we used three monolingual, single-speaker datasets simultaneously for the multi-speaker model, we did not use the language identity.

[3] https://github.com/kan-bayashi/ParallelWaveGAN

Table 3.3. Hyper-parameters and network architectures.

Feature extraction	
Sampling rate	22,050 Hz
Window size	46.4 ms (1,024 pt)
Shift size	11.6 ms (256 pt)
Acoustic feature	log-mel spectrogram 80 dim
Encoder	
# phoneme embedding dimension	512
# CNN layers	3
# CNN filters	512
CNN filter size	5
# BLSTM layers	1
# BLSTM units	512
Decoder	
# LSTM layers	2
# LSTM units	1024
# prenet layers	2
# prenet units	256
# postnet layers	5
# postnet filters	512
Postnet filter size	5
# Speaker embedding dimension	512
Attention	
# Dimensions in attention	128
# Filters in attention	32
Filter size in attention	31
Sigma in guided attention loss	0.4
Reduction factor (r)	1 (M_S) / 2 (M_M)
Optimization and minibatch	
Dropout rate	0.5
Zoneout rate	0.1
Learning rate	0.001
Optimization method	Adam with $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-6}$
Batch size	32 / 64
# Epochs	300 / 500 / 1000

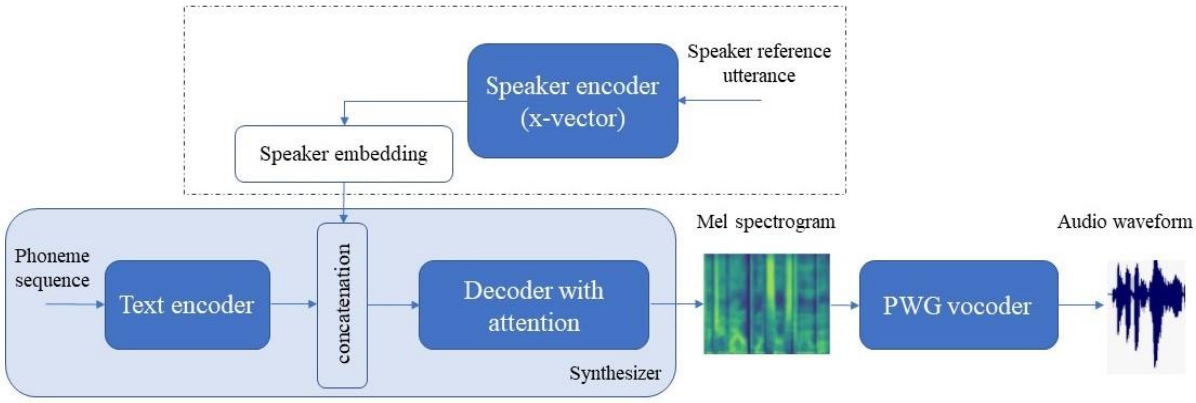

Figure 3.2. Overview of the base TTS system. Speaker embedding is used to train the multi-speaker model, but is not used for training the single-speaker model.

3.4 Methodology

3.4.1 Introduction

In this study, we used only 30 minutes of the target language data to train our proposed TTS model. However, this amount of data is not enough for training. In other words, when training the TTS model only 30 minutes of target language data from scratch, it is impossible to synthesize intelligible speech using the model. Therefore, the contribution of this study is the exploration of variants of methods used in our proposed low resource language TTS system. Furthermore, we compared the performance of models using each method described below with each other and with the baseline model. As a result, we explore the best-performing method, which reduces the gap between our low-resource model and the baseline M-MN model trained with a much larger amount (12 hours) of original target speech data. The following section describes each of these various methods we tested when building our proposed TTS system.

3.4.2 Description of each method

*3.4.2.1 **Method 1:** Cross-lingual transfer learning*

We trained the TTS model for our target language by transferring knowledge from our source languages in two ways. First, we used the source and target language datasets sequentially to

train the TTS model without speaker embedding. To obtain pre-trained models, we first trained the TTS models using only the English (E) or Japanese (J) source language datasets, each of which contains speech data from a different, single, female speaker. The English speech dataset is more than twice as long as the Japanese dataset. Therefore, in addition to model pre-trained with the entire English dataset (E24), we also pre-trained a model using randomly selected English speech data equal in size to the Japanese dataset (E10) to determine the effect of using different proportions of the high-resource languages. The model which was pre-trained with the entire English source language dataset (E24) was also adapted by training it again with the Japanese source language dataset, creating a fourth pre-trained model (EJ). These four TTS models (E10, E24, J and EJ), pre-trained with the high-resource language datasets, were trained again using the target language dataset, as in Tits et al. (2019), Bollepalli et al. (2019) and Chen et al. (2019), which in this study consisted of Mongolian language data. All of the datasets were recorded using the voice of a single male or female speaker. Therefore, we denote our single-speaker, sequentially-trained, cross-lingual transfer learning models as M_{SE10}-TL, M_{SE24}-TL, M_{SJ}-TL and M_{SEJ}-TL. The training flow diagrams for these models are shown in Figure 3.3.

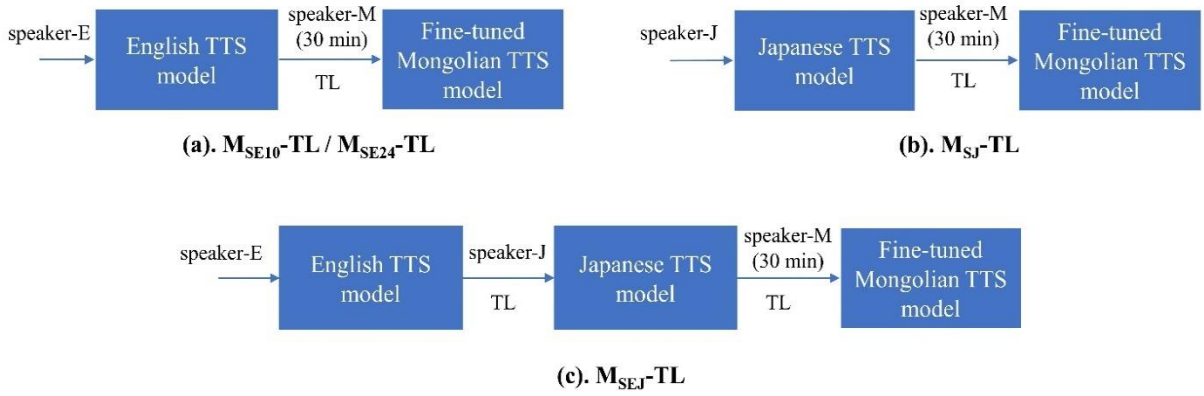

Figure 3.3. Training flow diagrams for our single-speaker TTS models. Transfer learning from the source languages to the target language is used, where (a) are models using only different amounts of the English dataset, (b) is a model using only the Japanese dataset, and (c) is a model using the entire datasets of both high-resource languages.

In order to evaluate the effectiveness of multi-speaker training, the source and target language datasets were also used to simultaneously train a second set of pre-trained TTS models with speaker embedding as the conditioned feature. In other words, we pre-trained three multi-speaker TTS models using bilingual datasets as follows: one using the Japanese source language dataset with the target language dataset; one using 10 hours of the English source language dataset with the target language dataset, and one using 24 hours of the English source language dataset with the target language dataset. One multi-speaker TTS model was also pre-trained using trilingual datasets (the entire, high-resource language datasets of both English and Japanese, along with 30 minutes of the target language dataset), with each dataset containing speech data from a different, single speaker. These four, pre-trained, multi-speaker TTS models were then fine-tuned using the same target language Mongolian dataset used to train the pre-trained multi-speaker models, as in Lee et al. (2020). The four multi-speaker, simultaneously-trained, cross-lingual transfer learning models were denoted as M_{ME10}-TL, M_{ME24}-TL, M_{MJ}-TL and M_{MEJ}-TL. The training flow diagrams for these models are shown in Figure 3.4.

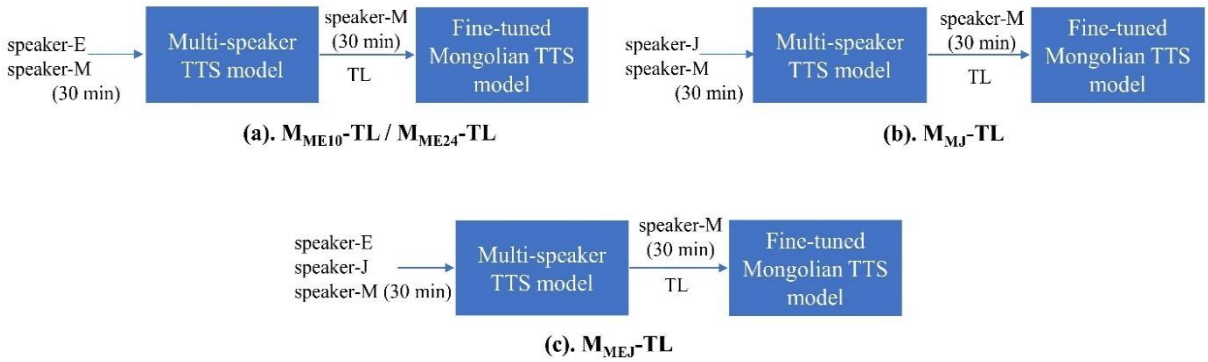

Figure 3.4. Training flow diagrams for our multi-speaker TTS models. Transfer learning from the source languages to the target language is used, where (a) are models using the different amounts of the English dataset and the Mongolian dataset, (b) is a model using the Japanese and Mongolian datasets, and (c) is a model using the entire datasets of both high-resource languages and the Mongolian dataset.

All pre-trained TTS models shown in Figures 3.3 and 3.4 were trained for 300 epochs, and the final models, fine-tuned with the target language dataset, were trained for 1,000 epochs. We

compared the performance of the four single-speaker and four multi-speaker models to understand how each training approach, i.e., using each high-resource language dataset separately, or both high-resource language datasets, either sequentially or simultaneously, affects the quality of the TTS system's output. We found that using both high-resource languages datasets simultaneously improved the performance of both the single-speaker and multi-speaker models. The results of our comparison are shown in Figures 3.11 and 3.12 in Section 3.5.2.2. Based on these results, we used both high-resource language datasets for model training when using the transfer learning method with data augmentation, as described in the following section.

*3.4.2.2 **Method 2:** Data augmentation*

Data augmentation is a method commonly used to address the problem of insufficient data. We used a basic audio data augmentation method which involves altering the pitch and speed of the original speech data, generating synthetic data from the original samples. We changed the pitch and speed of only 30 minutes of the original target language data using the SoX tool to synthetically generate a large amount of data with a wide range of variation, while using the same transcriptions as the original samples. The number of semitones of shift when changing the pitch was between -2.5 and 2.5, at steps of 0.5. The ratio of the speed of the augmented speech to the speed of the original speech was within the range of 0.7 to 1.55 times the speed of the original speech, at steps of 0.05, but no augmented data was generated at 1.05 times the original speed. The SoX tool shifts the full spectrum, not just the pitch, therefore all formants are also modified. We generated 26 different versions of 30 minutes of the original target language data, as shown in Table 3.4 of Section 3.4.2.4, creating a total of 13 hours of augmented target language data. We then trained a multi-speaker TTS model with both the augmented data and 30 minutes of the original target language data, treating it as a multi-speaker dataset. We also used the x-vectors for each virtual speaker generated during data augmentation. A single-speaker TTS model was also trained with the same data. We denoted these single-speaker and multi-speaker data augmentation models as M_S-DA and M_M-DA, respectively, and their training flow diagrams are shown in Figure 3.5. Both models were

trained with the augmented data for 500 epochs. The augmented data was also used to train the PWG neural vocoder, which was designated NV-DA.

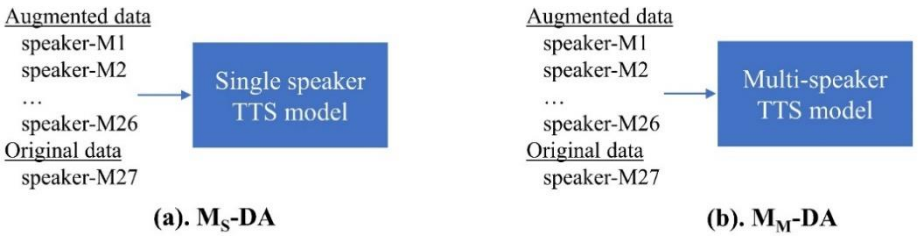

Figure 3.5. Method used to train TTS models with augmented target language data, where (a) is a single-speaker model, and (b) is a multi-speaker model.

3.4.2.3 ***Method 3:*** *Combination of cross-lingual transfer learning and data augmentation*

We then created two additional TTS models by training the single-speaker and multi-speaker models M_{SEJ}-TL-DA and M_{MEJ}-TL-DA with the two high-resource language datasets, the original target language dataset and the augmented data. These two models are almost the same as M_{SEJ}-TL and M_{MEJ}-TL, described in Section 3.4.2.1, except that augmented data was also used for training. The single-speaker model pre-trained using both high-resource languages datasets simultaneously was fine-tuned using augmented data and then fine-tuned again using the original target language data. The pre-trained multi-speaker model was trained using the trilingual datasets. During pre-training of the multi-speaker model, the two source language datasets are single-speaker datasets, while the target language dataset contains both original and augmented target data, thus it can be considered a multi-speaker dataset with 27 "speakers". The pre-trained multi-speaker TTS model was then fine-tuned using the original target language dataset.

Both the pre-trained single-speaker and pre-trained multi-speaker models were trained for 300 epochs using the high-resource language datasets, and the pre-trained single-speaker model was also fine-tuned by training it with augmented data for 500 epochs. The final models were both fine-tuned by training each model for 1,000 epochs using the original target language dataset. Training flow diagrams for the single-speaker and multi-speaker models are shown in Figure 3.6.

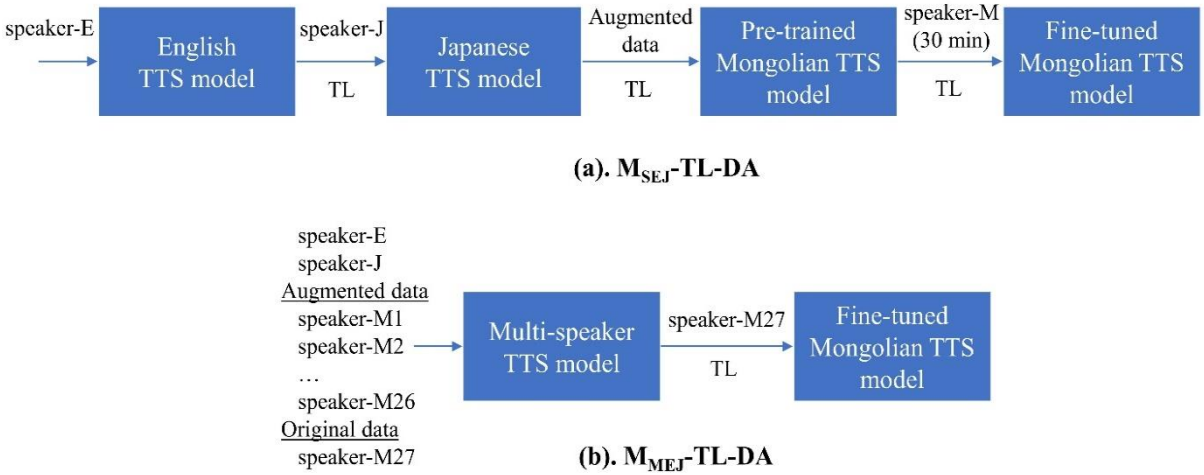

Figure 3.6. Methods used to train TTS models using cross-lingual transfer learning and augmented data, where (a) is a single-speaker model, and (b) is a multi-speaker model.

*3.4.2.4 **Method 4:** Combination of cross-lingual transfer learning and data augmentation with additional finetuning*

For these TTS models, we added additional fine-tuning steps, using some of the augmented target language data to further improve the models' gradual adaptation to the target language. We used t-SNE (Maaten and Hinton 2008) to visualize the x-vectors extracted from the real and virtual speakers' speech, as shown in Figure 3.7. Then we divided the augmented data into three sets based on this visualization, as shown in Figure 3.8. Table 3.4 shows the identity of each virtual speaker generated by changing the pitch and speed factors of the original speech data, where the identity of the real speaker is 27. The x-vectors extracted from virtual speakers 1, 2, 11, 12, 13, 14, 15, 21, 22, 23, 24, 25 and 26 were judged to be farther away from the x-vectors of the real speaker. Therefore, the first set of augmented data contained these 13 copies of the 30 minutes of the original target language data, which sounded very different from the original target language speech. The second set of augmented data contained 7 copies (from virtual speakers 3, 9, 10, 17, 18, 19 and 20) of the original data which sounded more similar to the original target speaker's voice than the augmented speech in the first set. The x-vectors extracted from virtual speakers 4, 5, 6, 7, 8 and 16 were closest to the x-vectors of the real speaker. Therefore, the third set of augmented data contained these 6 copies of the original data, which sounded the most similar to the target speaker's actual voice. We used these three sets of the augmented data to fine-tune the pre-trained model sequentially. The single speaker

model, which was pre-trained with the two high-resource language datasets, was then sequentially fine-tuned using the same three sets of augmented data. Finally, the pre-trained, single-speaker model was fine-tuned with the original target language dataset. For the multi-speaker model, the pre-trained multi-speaker model was trained with the trilingual datasets (two high-resource language datasets plus the original and augmented target language datasets) and then sequentially fine-tuned using the three sets of augmented data. The model was then fine-tuned again using the original target language dataset. We denoted these single-speaker and multi-speaker models as M_{SEJ}-TL-DA_D and M_{MEJ}-TL-DA_D, respectively. Training flow diagrams for these models are shown in Figure 3.9. Both the pre-trained single-speaker and pre-trained multi-speaker models were trained for 300 epochs using the high-resource language datasets. We then further trained both of these fine-tuned models for 500 epochs at each fine-tuning step, using the sets of augmented data sequentially, before a final 500 epochs of training using the original target language data.

Table 3.4. Number of semitones of pitch shift (PF), or ratio of speed of the new speech to speed of the original speech (SF), used when generating augmented data from the original data, for each virtual speaker.

Speaker #	Pitch or speed factor (PF / SF)		Speaker #	Pitch or speed factor (PF /SF)	
1	-2.5	***PF***	**14**	0.85	***SF***
2	-2.0		**15**	0.9	
3	-1.5		**16**	0.95	
4	-1.0		**17**	1.1	
5	-0.5		**18**	1.15	
6	0.5		**19**	1.2	
7	1.0		**20**	1.25	
8	1.5		**21**	1.3	
9	2.0		**22**	1.35	
10	2.5		**23**	1.4	
11	0.7	***SF***	**24**	1.45	
12	0.75		**25**	1.5	
13	0.8		**26**	1.55	

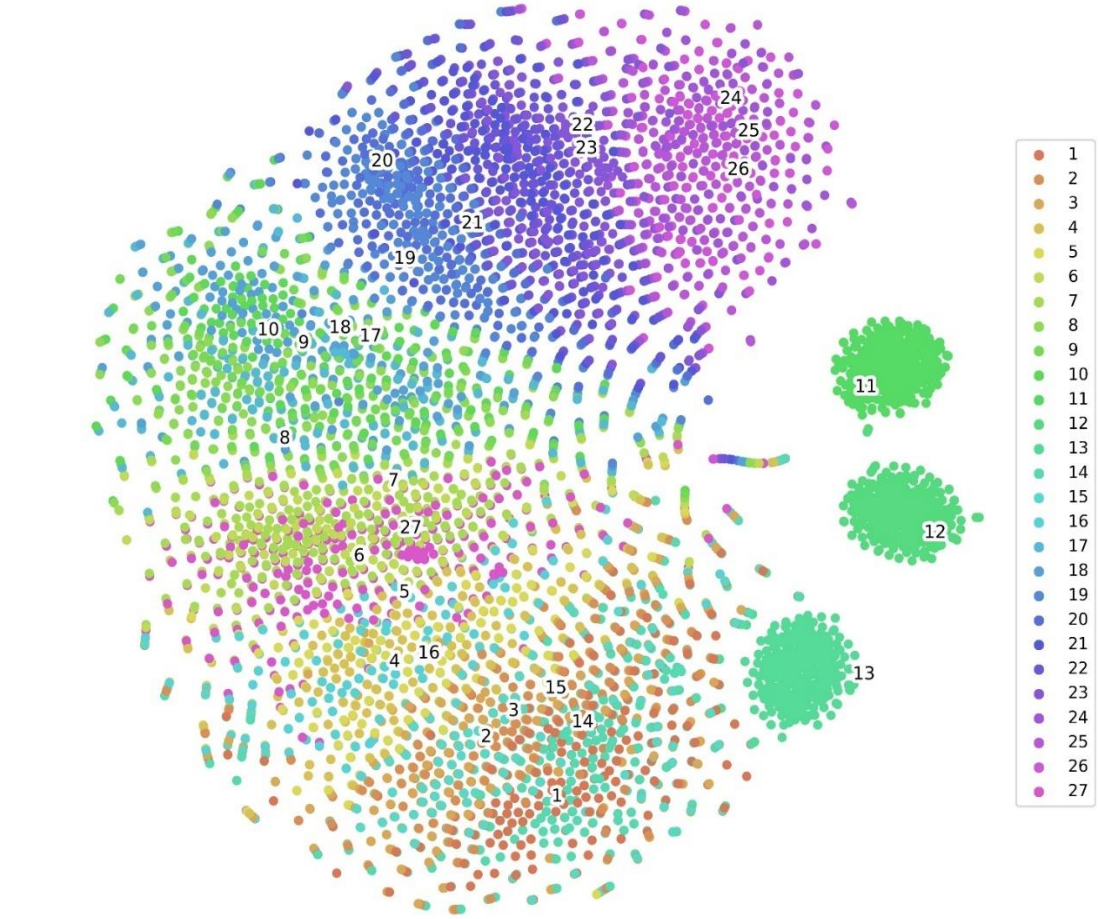

Figure 3.7. t-SNE visualization of x-vectors extracted from the speech of the real and virtual speakers, where Speaker 31 is the real speaker.

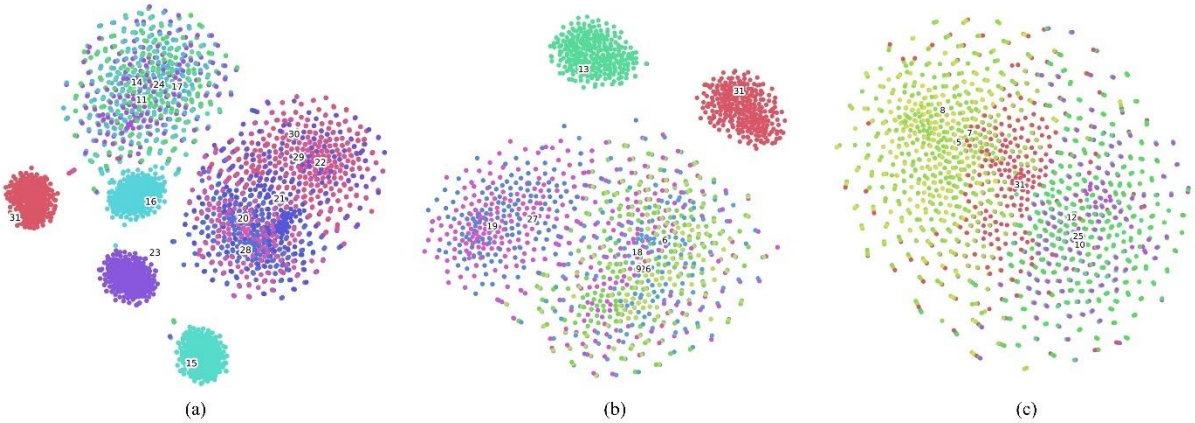

Figure 3.8. t-SNE visualization of x-vectors extracted from the speech data when divided into three sets, where (a), (b) and (c) represent the first, second and third sets, respectively.

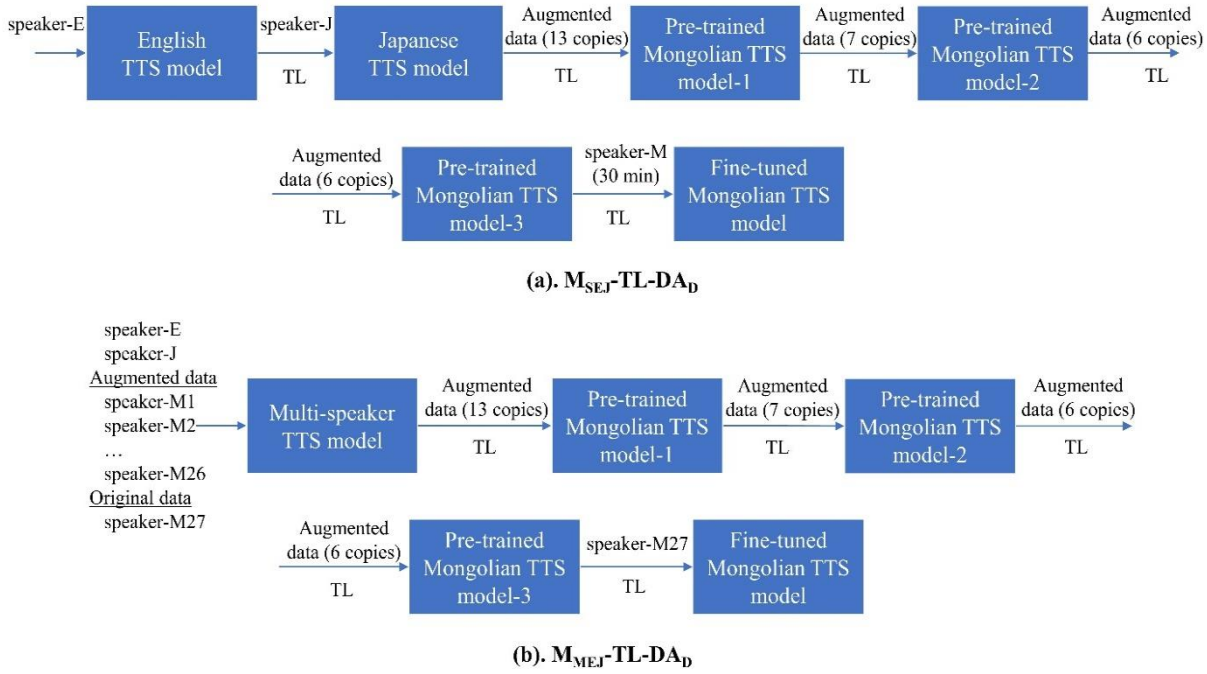

Figure 3.9. Methods used to train TTS models using cross-lingual transfer learning and augmented data with additional fine-tuning steps, where (a) is a single speaker model, and (b) is a multi-speaker model.

All proposed systems described in Section 3.4.2 are summarized in Table 3.7 at the end of Section 3.5.

3.5 Implementation

3.5.1 Evaluation

We conducted an AB preference test to assess the quality of the output from the neural vocoders trained using the original target data (NV-MN) and augmented target data (NV-DA). Our subjects were asked to select the higher quality speech when comparing 15 speech samples generated by each vocoder. The results of this evaluation are shown in Table 3.6.

For the spectrogram prediction models, we conducted subjective naturalness and speaker similarity tests (Test-1 to Test-5). To evaluate the naturalness of the synthesized speech produced when using each TTS model, we conducted subjective tests using eight speech samples produced by each model which were not contained in the training dataset. We used the

web-based MUltiple Stimuli with Hidden Reference and Anchor (webMUSHRA) test (Schoeffler et al. 2018) to evaluate naturalness. Figure 3.10 shows a screenshot of MUSHRA listening test designed with webMUSHRA. All of the speech samples being evaluated are presented in one panel, and the samples within the panel are randomized. We created four separate naturalness test sets (Test-1, Test-2, Test-3 and Test-4, shown in Table 3.5), each containing eight stimulus panels. Each panel included a hidden reference and hidden anchors. In addition to the hidden reference and hidden anchors, Table 3.5 also shows all the systems that generated the speech samples included in each stimulus panel. Test-1 and Test-2 are naturalness evaluation tests of the single-speaker and multi-speaker models used in the cross-lingual transfer learning method explained in Section 3.4.2.1, in order to investigate the effect of the high resource language dataset. Test-3 is a comparison of all of the proposed single-speaker and multi-speaker models described in Section 3.4.2, conducted to determine the best performing method. Test-4, the final naturalness evaluation test, was conducted to compare the output of the models when using the best performing method (a combination of cross-lingual transfer learning and data augmentation, as described in Section 3.4.2.3), when different amounts of the target language data were used during training. All of the TTS systems shown in Table 3.5 are summarized in Table 3.7 at the end of Section 3.5. The results of these naturalness evaluations are shown in Figures 3.11, 3.12, 3.13 and 3.14 of Section 3.5.2.2.

Figure 3.10. MUSHRA listening test GUI

Table 3.5. Systems used to generate the speech samples included in each stimulus panel of the MUSHRA subjective naturalness tests

Tests / Systems	MUSHRA subjective naturalness tests			
	Test-1	**Test-2**	**Test-3**	**Test-4**
Systems	- M_{SJ}-TL - M_{SE10}-TL - M_{SE24}-TL - M_{SEJ}-TL	- M_{MJ}-TL - M_{ME10}-TL - M_{ME24}-TL - M_{MEJ}-TL	- M_{M}-DA - M_{SEJ}-TL - M_{SEJ}-TL-DA - M_{SEJ}-TL-DA_{D} - M_{MEJ}-TL - M_{MEJ}-TL-DA - M_{MEJ}-TL-DA_{D} - M-MN	- M_{MEJ}-TL-DA - M_{MEJ}-TL-DA_{1hour} - M_{MEJ}-TL-DA_{2hours} - M_{MEJ}-TL-DA_{3hours} - M-MN
Hidden reference	- Ground truth	- Ground truth	- Ground truth	- Ground truth
Hidden anchors	- M_{SEJ}-TL-DA - M-MN	- M_{MEJ}-TL-DA - M-MN	- M_{MEJ}-TL-DA_{3hours}	- M_{SEJ}-TL - M_{MEJ}-TL

A MUSHRA speaker similarity evaluation (Test-5) was also performed to compare the output of proposed multi-speaker model M_{MEJ}-TL-DA, which uses a combination of transfer learning and data augmentation, with the ground truth Mongolian target speech data. Study participants also compared the output of the baseline M-MN model with the ground truth. They evaluated the similarity of eight speech samples generated from each of these two models, in comparison to the ground truth speech data, to assess their similarity to the original target language speech. The results of these comparisons are shown in Figure 3.15 of Section 3.5.2.2. These comparisons were performed because, in addition to the small, target language dataset, two high-resource language datasets and augmented data were also used to build the basic single-speaker TTS system used in the proposed model.

Twenty-two subjects were asked to rate the naturalness and speaker similarity of the synthesized audio, and twenty-nine subjects were asked to rate the quality of the output from the neural vocoders. All of the subjects who participated in the subjective naturalness, similarity

and quality tests were native Mongolian speakers. Speech samples generated by each of these models and vocoders are publicly available.[4]

3.5.2 Results

In the first test, we evaluated the PWG neural vocoder trained with augmented data, which was used to generate the waveform in all of our experiments. In the second test, we investigated the effects of using the cross-lingual transfer learning method with the high-resource languages on low-resource language TTS performance. We found using both high-resource language datasets improved the performance of both the single-speaker and multi-speaker models. Therefore, we used this training strategy in the following experiments. We compared all of our proposed models with the baseline model and ground truth in the third test, to determine which of the proposed models was able to achieve the best performance, which was the multi-speaker model utilizing a combination of cross-lingual transfer learning and data augmentation with additional fine-tuning. Although the proposed multi-speaker model with additional fine-tuning achieved the best performance, we chose the proposed multi-speaker model utilizing a combination of cross-lingual transfer learning and data augmentation for the next two tests because it is less time-consuming to train and has almost similar performance to the best-performing model. In the fourth test, we trained it with larger amounts of target language data to investigate the amount of the target language training data needed to obtain a model with the same or similar performance as the baseline model. In the final test, we evaluated the speaker similarity of speech samples generated by the model selected in previous test. In Table 3.7, at the end of Section 3.5, we summarize all of the systems evaluated in this study.

3.5.2.1 PWG neural vocoder

In this study we proposed a TTS system containing both a spectrogram prediction network and a neural vocoder, for use when only a small amount of target data is available. To evaluate the effectiveness of training the vocoder with augmented data, we trained a PWG neural vocoder with 13 hours of our augmented data and thirty minutes of original target language data (NV-DA), while the baseline vocoder was trained with 12 hours of original target language data

[4] https://zolzaya-byambadorj.github.io/tts/

(NV-MN). We then performed an AB preference test to compare the output of the two PWG vocoders, as evaluated by twenty-nine, native Mongolian speaking subjects. We used the same M-MN baseline model used by the spectrogram prediction model for both of the vocoders. Listeners had the option of selecting "*no preference*" if the difference between the synthesized speech pairs was too difficult to distinguish. The test results in Table 3.6 show that the quality of the synthesized speech generated by the two vocoders was almost the same, as it was difficult for the listeners to distinguish the difference. Therefore, the PWG neural vocoder trained with augmented data was used to generate the waveform in all of the following experiments investigating the best method of training the spectrogram prediction network, as described in Section 3.4.2.

Table 3.6. Results of AB preference test on vocoders trained with original (NV-MN) and augmented (NV-DA) data.

NV-MN (baseline)	NV-DA	No preference
21.61%	16.09%	**62.30%**

3.5.2.2 Spectrogram prediction models

- ***Test-1 and Test-2:*** *Cross-lingual transfer learning*

  In these experiments, we investigated the effects of training the models with high-resource languages on low-resource language TTS performance. Figures 3.11 and 3.12 shows the boxplots of the MUSHRA subjective naturalness scores for the single-speaker and multi-speaker TTS models, respectively, described in Section 3.4.2.1. Native Mongolian speaking subjects performed these naturalness evaluations. As we expected, when using the same amount of data from each of the high-resource languages, the effect of Japanese language training on low-resource target language TTS performance was more beneficial than English language training for both the single-speaker and multi-speaker models. We think this is because, as explained in Section 3.2, Japanese and Mongolian are more similar than English and Mongolian. However, we can see that when the entire English language dataset was used for training, the performance of both the single-speaker and multi-speaker models was

better than when using the Japanese high-resource language dataset. A reason for this could be the size of the datasets. The English speech dataset is more than twice as long as the Japanese dataset. Using both high-resource language datasets improved the performance of both the single-speaker and multi-speaker models more than using only one high-resource language dataset. Therefore, we used both high-resource language datasets to train the single-speaker and multi-speaker models using the transfer learning method in the rest of the experiments. In addition, as shown in Figures 3.11 and 3.12, the performance of the multi-speaker models (M_{MJ}-TL, M_{ME10}-TL, M_{ME24}-TL and M_{MEJ}-TL) was better than the performance of the corresponding single-speaker models (M_{SJ}-TL, M_{SE10}-TL, M_{SE24}-TL and M_{SEJ}-TL), even though MUSHRA naturalness evaluations were conducted separately. Among the multi-speaker models trained with only one high-resource language dataset during the pre-training stage (M_{MJ}-TL, M_{ME10}-TL and M_{ME24}-TL), when the target language dataset was not used in the pre-training stage, the operation of the models was almost the same as that of the corresponding single-speaker models, except for the use of speaker embedding. This suggests that using target language data when training the pre-trained model improves the performance of the TTS model. Note that we used only 30 minutes of the target language data to train the models shown in Figures 3.11 and 3.12. The results for these models are low because the use of only 30 minutes of target language data for pre-training and fine-tuning the models is insufficient for generating good quality speech when using cross-lingual transfer learning. Studies Tits et al. (2019) and Lee et al. (2020) also showed that the amount of target data used affects the performance of the final fine-tuned model. But we can see from these experiments that the use of high-resource languages helps the models learn to synthesize speech in the low-resource target language.

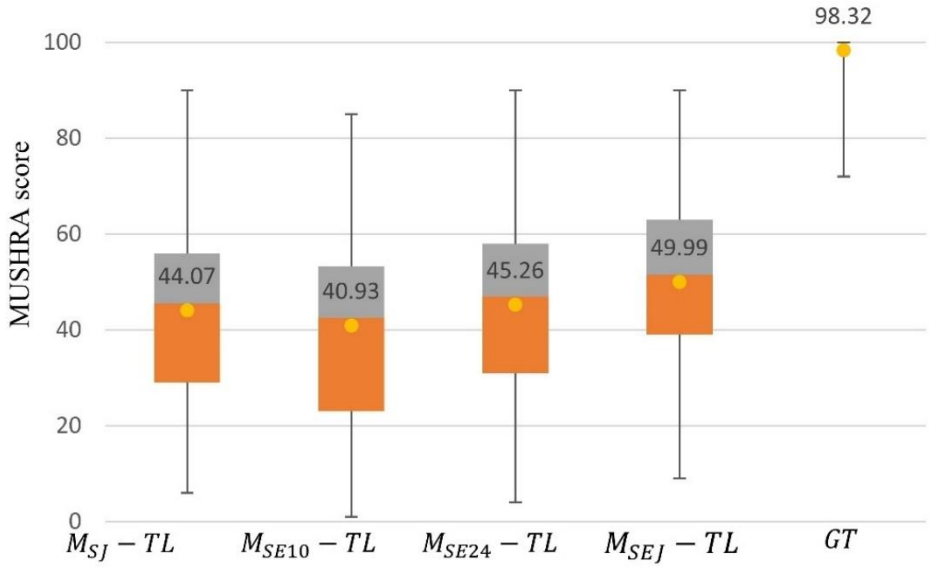

Figure 3.11. MUSHRA naturalness scores for single-speaker models trained using cross-lingual transfer learning (one high-resource language or both, $M_{SJ\ (Japanese)}$, $M_{SE10\ (English,\ 10\ hours)}$, $M_{SE24\ (English,\ 24\ hours)}$, $M_{SEJ\ (English\ and\ Japanese)}$: sequentially trained single-speaker models)

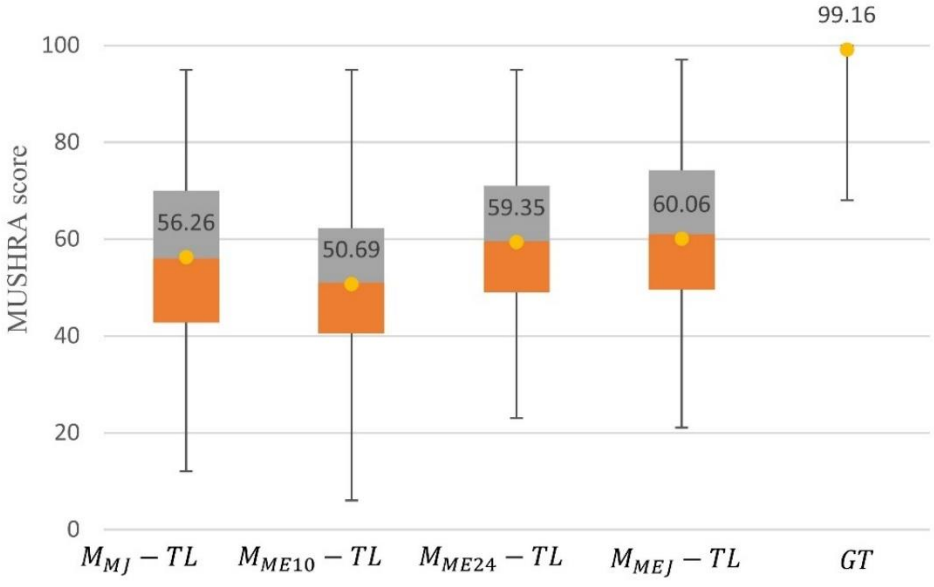

Figure 3.12. MUSHRA naturalness scores for multi-speaker models trained using cross-lingual transfer learning (one high-resource language or both, $M_{MJ\ (Japanese)}$, $M_{ME10\ (English,\ 10\ hours)}$, $M_{ME24\ (English,\ 24\ hours)}$, $M_{MEJ\ (English\ and\ Japanese)}$: simultaneously trained multi-speaker models)

- ***Test-3:*** *All proposed methods*

  Figure 3.13 shows the boxplots of the MUSHRA subjective naturalness scores, as rated by native Mongolian speakers, for all of the proposed single-speaker and multi-speaker models described in Section 3.4.2. As discussed in Test-1 and Test-2, we found that using both high-resource language datasets simultaneously improved the performance of both the single-speaker and multi-speaker models. Therefore, we used both high-resource language datasets for model training when using the transfer learning method with data augmentation, described in Sections 3.4.2.3 and 3.4.2.4. All of the proposed models evaluated in Figure 3.13 were trained using only 30 minutes of original target language data.

  The amount of augmented data (DA) used for training these models is almost same as the amount of original target language training data used for the baseline single-speaker model M-MN, which achieved the best results in our experiment. The augmented data was used to train both single-speaker and multi-speaker TTS models, but the single-speaker model (M_S-DA) was a failure because it could not synthesize intelligible speech. Therefore, we did not ask the study participants to rate this model. Although the naturalness score of the multi-speaker model trained with augmented data (M_M-DA) was lower than the scores of most of the other models, it was able to learn how to synthesize intelligible speech using only augmented data and 30 minutes of the original target language data. This suggests that adding speakers could improve training of multi-speaker models, since the augmented data can be considered to be multi-speaker data. Also note that although the M_{SEJ}-TL model was trained using both high-resource language datasets and the original target speech data, the M_M-DA model received a higher score, even though the M_M-DA model was unable to learn the pronunciations of some of the letters which appeared very few times in the 30 minutes of original target language data, which was used to generate the augmented data. For example, the letters 'ц,' 'к,' 'ф' and 'п', which could not be synthesized by the M_M-DA model, occurred less than 100 times in the 30 minutes of the target language data. But some letters, such as 'ъ' and 'я,' which also occurred less than 100 times in the data, could be learned by this model. This is because the phonetic notations of the letters 'ю,' 'я,' 'ё' are a

combination of phonemes. Although these letters appear infrequently in the 30 minutes of original target language data before the transcriptions were converted into their phonetic representations, each phoneme contained in the phoneme notations of these letters occurred more than 100 times in the 30 minutes of original target language data. Furthermore, the phonetic notation of some letters, such as 'ъ,' is the same as the phonetic notation of some other letters, such as 'ь,' 'ы,' 'й' and 'и', because these letters have the same pronunciation. Therefore, although these letters only occurred a few times in the data, since we used phonetic representations the pronunciations of these letters could still be learned. We also observed that some letters which occurred less than 200 times in the data could not be synthesized clearly. However, if the transcript to be converted into speech does not include these particular, low-frequency letters, the synthesized speech created using the M_M-DA model sounds very reasonable. Thus, in general, the performance of the multi-speaker model using augmented data (M_M-DA) shows that the use of augmented data can improve the performance of TTS models.

Regarding the models trained using the cross-lingual transfer learning method, the pronunciations of the letters that only occurred a few times in the target language data could be learned from the high-resource language datasets, since there are overlapping phonemes in the source and target languages. Therefore, learned phoneme embeddings are shared by the different languages. The model trained with only 30 minutes of target language data from scratch (M-MN_{30}) could not synthesize intelligible speech. However, the performance of the single-speaker and multi-speaker transfer learning models (M_{SEJ}-TL and M_{MEJ}-TL) shows that the cross-lingual transfer learning approach improves TTS model performance when only a small amount of target data is available. On the other hand, single-speaker model M_{SEJ}-TL was trained using data from three languages sequentially, while multi-speaker model M_{MEJ}-TL was trained using data from three languages simultaneously. As a result, the naturalness score of multi-speaker model M_{MEJ}-TL is higher than that of single-speaker model M_{SEJ}-TL. This suggests that adding languages could also improve the training of multi-speaker models.

Each of the proposed methods, i.e., using only transfer learning or only data augmentation, were capable of improving the performance of the TTS model. Therefore,

unsurprisingly, we can also see in Figure 3.13 that a combination of both the transfer learning and data augmentation methods improved both single-speaker (M_{SEJ}-TL-DA) and multi-speaker (M_{MEJ}-TL-DA) model performance. As mentioned previously, adding speakers or adding languages each improved the performance of the multi-speaker models. In the case of the model M_{MEJ}-TL-DA, we added both languages and speakers simultaneously. As a result, the performance of the multi-speaker model with data augmentation (M_{MEJ}-TL-DA) was superior to that of the single-speaker model with data augmentation (M_{SEJ}-TL-DA), the multi-speaker model without data augmentation (M_{MEJ}-TL) and the multi-speaker, single-language model with data augmentation (M_{M}-DA). We can also see that the performance of TTS models M_{SEJ}-TL-DA_D and M_{MEJ}-TL-DA_D improved slightly when fine-tuning steps that included the use of augmented data were added. Related works (Ko et al. 2015; Cooper et al. 2020; Liu et al. 2020) have used data augmentation-generated synthetic speech created by changing the speed and tempo of the original speech within a relatively narrower range of variation, compared to the augmentation method used in our study. In other words, the differences between the synthetic and real data used in previous studies were not as great as in our approach. In contrast, we generated our synthetic speech using a wider range of variation, and 26 versions of the data were generated from the original speech. As a result of this wider variation, the speech of some of the virtual speakers is very different from the speech of the real speaker, while some is very similar to the real speaker's speech. Therefore, gradual fine-tuning as part of a multi-stage process may yield further gains in performance. On the other hand, although the naturalness score of single-speaker model M_{SEJ}-TL-DA_D is lower than that of multi-speaker model M_{MEJ}-TL-DA_D, we observed that the effect of the additional fine-tuning steps using augmented data was greater on the single-speaker model than on the multi-speaker model, when their naturalness scores are compared with those of the corresponding single- and multi-speaker models M_{SEJ}-TL-DA and M_{MEJ}-TL-DA. We suspect this may occur because the single-speaker model 'discovers' each new speaker at each training stage, while the multi-speaker model encounters all of the speakers during the first training stage, thus

gradual fine-tuning may have been more effective for the single-speaker model and less effective for the multi-speaker model.

Finally, the results shown in Figure 3.13 indicate that the performance of the multi-speaker (M_M/M_{MEJ}) TTS models was superior to that of the single speaker (M_S/M_{SEJ}) TTS models. In other words, multi-speaker models were effective as intermediate models when constructing a single-speaker, low-resource TTS model. The score of the proposed M_{MEJ}-TL-DA_D model was higher than the scores of the other models trained with a limited amount of target language data, but lower than the score of the baseline M-MN model.

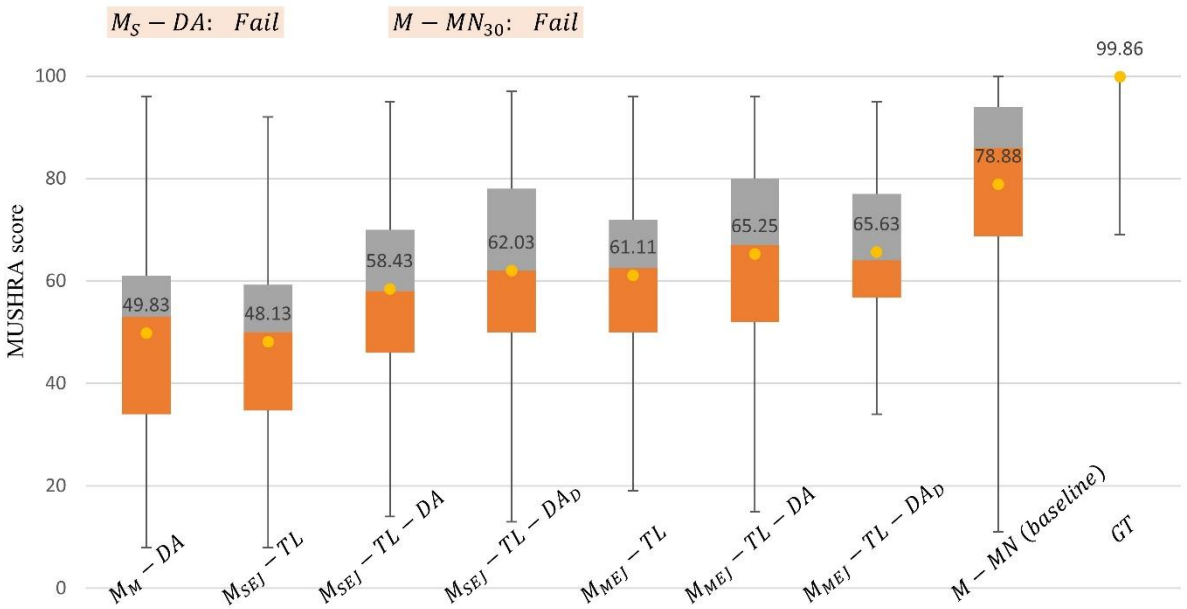

Figure 3.13. MUSHRA naturalness scores for all single-speaker and multi-speaker models.

M-MN = TTS model trained with 12 hours of target language data
$M\text{-}MN_{30}$ = TTS model trained from scratch with only 30 minutes of target language data
M_{SEJ} = sequentially trained single-speaker model
M_{MEJ} = simultaneously trained multi-speaker model
TL = cross-lingual transfer learning
DA = data augmentation
DA_D = data augmentation method with additional fine-tuning

- ***Test-4:*** *Size of target language training data*

  We also wanted to know the minimum amount of original target language training data that was needed to obtain a model with the same performance as the baseline model. Therefore, we increased the 30 minutes of original target language data to 1, 2 or 3 hours of training data. We selected this data randomly, and then created 26 different versions of augmented data using the same amounts of original target language data (1, 2 or 3 hours) as described above. Although the proposed multi-speaker model with additional fine-tuning (M_{MEJ}-TL-DA_D in Figure 3.13) achieved the best performance, we chose the proposed multi-speaker model utilizing a combination of cross-lingual transfer learning and data augmentation (the M_{MEJ}-TL-DA model described in Section 3.4.2.3) because it is less time-consuming to train and has almost similar performance to the best-performing model. We trained it with these various amounts of target language data, and with the additional augmented training data created using this extra target language data. The performance of these variously trained models, and the baseline model, were then compared based on the naturalness of the output speech, which was measured using a MUSHRA test. Figure 3.14 shows the boxplots of the naturalness scores for these models. The performance of the models improved as the amount of original target language data increased. Three hours of target language data were sufficient to cover variations in pronunciation, and fluctuations in the speakers' voices were enhanced using data augmentation. Furthermore, the multi-speaker model was able to capture the features of the original voices more accurately than the single-speaker model. Therefore, the naturalness score of the proposed model trained with three hours of the original target language data was similar to the score of the baseline model trained with 12 hours of target language data. We also observed that the proposed models trained with two and three hours of the original target language data synthesized very clear, good quality speech, while the baseline model synthesized slightly more nuanced speech. Therefore, the native Mongolian speaking subjects may have preferred the output of the baseline model.

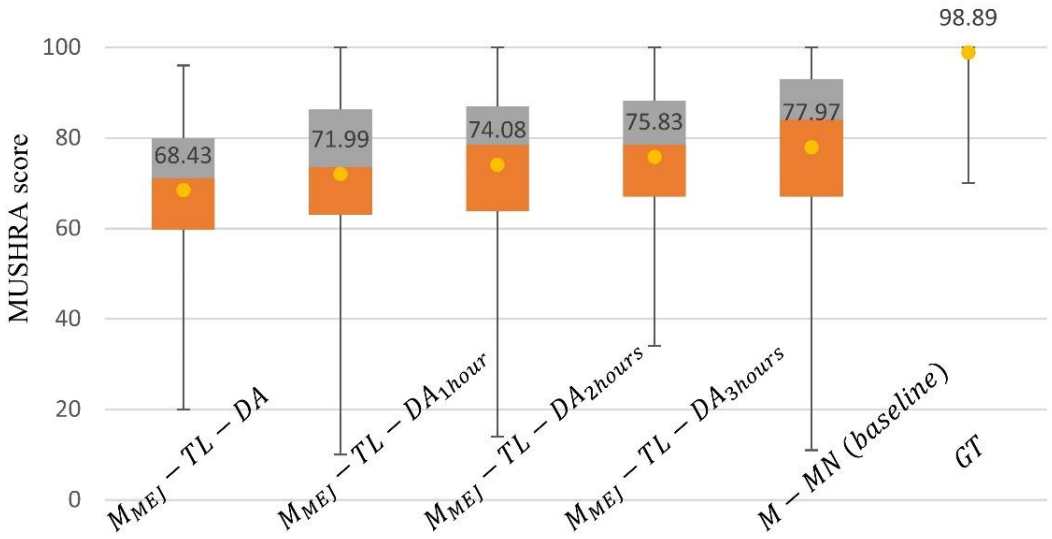

Figure 3.14. MUSHRA naturalness scores for baseline and proposed multi-speaker model trained with various amounts of target language data.

- ***Test-5:*** *Speaker similarity*

  A MUSHRA speaker similarity evaluation was performed on the output of the M_{MEJ}-TL-DA and baseline models. Note that we again chose the proposed multi-speaker model M_{MEJ}-TL-DA, which utilizes a combination of cross-lingual transfer learning and data augmentation (as described in Section 3.4.2.3), instead of the best performing model M_{MEJ}-TL-$DA_{D,}$ for our speaker similarity evaluation test. We asked our native Mongolian speaking subjects to evaluate speech samples generated by our proposed and the baseline models in comparison to the ground truth of the original Mongolian speaker. The subjects were asked to, "Please rate the speaker similarity of each speech sample in comparison to the reference sample, on a scale of between 0 (definitely different) to 100 (definitely the same)." The results are shown in Figure 3.15. Our goal in this study was to obtain a model whose performance is the same or similar to that of the baseline model in a low resource scenario. The baseline model was trained with more than 10 hours of target language data, while our best performing proposed models fine-tuned pre-trained models trained with two high-resource languages and augmented data. Therefore, we wanted to know how training TTS model with the high-resource language data and augmented data affect speaker similarity between the speech samples

generated by our proposed model and the ground truth. The results of our evaluation show that speaker similarity of the speech samples generated by our proposed model to the ground truth was slightly lower when using cross-lingual training and augmented data. But the similarity score of our proposed model was only slightly lower than the similarity score of the baseline model, despite the proposed model using far less original target language data for training.

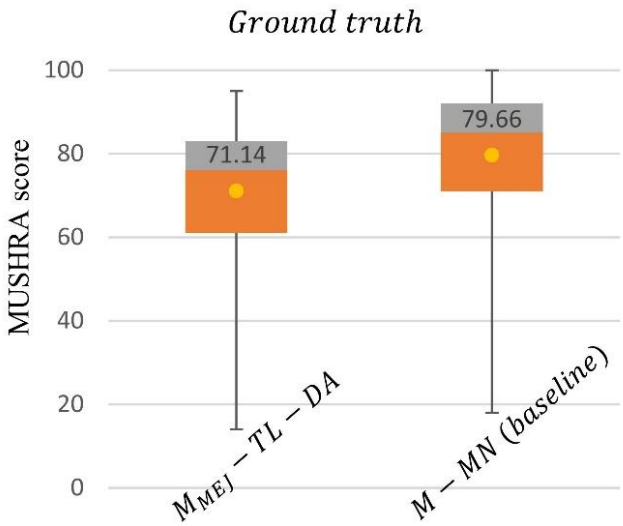

Figure 3.15. Speaker similarity comparison of speech samples generated using our proposed model and using the baseline model, in relation to the ground truth.

Finally, we have summarized all of systems evaluated in this study in Table 3.7.

Table 3.7. Summarization of all systems tested.

#	System	Training stage 1	Training stage 2	Training stage 3	Training stage 4	Training stage 5	Training stage 6
Spectrogram prediction models							
1	M-MN (baseline)	MN12h	-	-	-	-	-
2	M_{SJ}-TL	JP	MN30	-	-	-	-
3	M_{SE10}-TL	EN10	MN30	-	-	-	-
4	M_{SE24}-TL	EN24	MN30	-	-	-	-
5	M_{SEJ}-TL	EN24	JP	MN30	-	-	-
6	M_{S}-DA	AD_{30} + MN30	-	-	-	-	-
7	M_{SEJ}-TL-DA	EN24	JP	AD_{30}	MN30	-	-
8	M_{SEJ}-TL-DA_{D}	EN24	JP	AD_{30}-set1	AD_{30}-set2	AD_{30}-set3	MN30
10	M_{MJ}-TL	JP + MN30	MN30	-	-	-	-
12	M_{ME10}-TL	EN10 + MN30	MN30	-	-	-	-
14	M_{ME24}-TL	EN24 + MN30	MN30	-	-	-	-
15	M_{MEJ}-TL	EN24+JP + MN30	MN30	-	-	-	-
16	M_{M}-DA	AD_{30} + MN30	-	-	-	-	-
17	M_{MEJ}-TL-DA	EN24+JP + AD_{30} + MN30	MN30	-	-	-	-
18	M_{MEJ}-TL-DA_{D}	EN24+JP + AD_{30} + MN30	AD_{30}-set1	AD_{30}-set2	AD_{30}-set3	MN30	-
19	M_{MEJ}-TL-$DA_{1\ hour}$	EN24+JP + AD_{1h} + MN1h	MN1h	-	-	-	-
20	M_{MEJ}-TL-DA_{2hous}	EN24+JP + AD_{2h} + MN2h	MN2h	-	-	-	-
21	M_{MEJ}-TL-DA_{3hous}	EN24+JP + AD_{3h} + MN3h	MN3h	-	-	-	-
Neural vocoders							
22	NV-MN (baseline)	MN12h	-	-	-	-	-
23	NV-DA	AD_{30} + MN30	-	-	-	-	-

Model type

M_{SXXX} - Single-speaker TTS model

M_{MXXX} - Multi-speaker TTS model

NV – neural vocoder

Method used for model training

TL - Cross-lingual transfer learning

DA - Data augmentation

TL-DA - Cross-lingual transfer learning and data augmentation
$TL\text{-}DA_D$ - Cross-lingual transfer learning and data augmentation with additional fine tuning

Databases used for training stages

EN10 - 10 hours of the English dataset
EN24 - 24 hours of the English dataset
JP - 10 hours of the Japanese dataset
MN12h – 12 hours of the target language dataset
MN30 - 30 minutes of the target language dataset
MN1h - 1 hour of the target language dataset
MN2h - 2 hours of the target language dataset
MN3h - 3 hours of the target language dataset
AD_{30} - augmented data generated from 30 minutes of the target language dataset
AD_{30}-set1 - first set of augmented data generated from 30 minutes of the target language dataset
AD_{30}-set2 - second set of augmented data generated from 30 minutes of the target language dataset
AD_{30}-set3 - third set of augmented data generated from 30 minutes of the target language dataset
AD_{1h} - augmented data generated from 1 hour of the target language dataset
AD_{2h} - augmented data generated from 2 hours of the target language dataset
AD_{3h} - augmented data generated from 3 hours of the target language dataset

3.6 Conclusion

In this study we proposed a TTS system containing both a spectrogram prediction network and a neural vocoder, for use when only a small amount of target data is available. We compare the performance of various TTS models and found that multi-speaker models were effective as intermediate models when constructing a single-speaker, low-resource TTS model. We trained some models using only transfer learning and some using only data augmentation, to evaluate how each method affected the naturalness of the speech output by the TTS model. We found that training the TTS model using both cross-lingual transfer learning and data augmentation improved performance, reducing the gap between our low-resource model and the baseline M-MN model, which was trained with a much larger amount (12 hours) of original target speech data. We then tried adding additional fine-tuning steps using augmented data and the original target language data, which slightly improved the performance of our proposed model.

Although the naturalness and speaker similarity scores for our proposed model using both cross-lingual transfer learning and data augmentation was very reasonable, we also investigated increasing the amount of original target language data used for training. By increasing the amount of original target language data used for model training from 30 minutes to 3 hours, our proposed model using both cross-lingual transfer learning and data augmentation achieved performance very close to that of the baseline model.

We also trained the PWG vocoder using augmented data generated from 30 minutes of the original target language data. As a result, our proposed method achieved almost the same speech quality as the vocoder trained with the entire 12 hours of target language data. As a result, our proposed TTS system, consisting of a spectrogram prediction network and a PWG neural vocoder, was able to achieve almost equivalent performance to the baseline model using only 3 hours of original target language training data, and reasonable performance using only 30 minutes of original target language training data.

Chapter 4: Conclusion and Future work

4.1 Conclusion

In this study, we aimed to build a Mongolian TTS system, which generates speech from canonical and noisy, transliterated text. To make our proposed TTS system, we performed two main parts, text normalization, and speech synthesis. Therefore, as for the Mongolian language, the first issue we discussed is social media text normalization, which is an important preprocessing for our proposed TTS system. The main problem to be solved in this study is normalizing OOV words that are not contained in the training data. We first identified the challenges of normalizing the noisy, transliterated Mongolian words. We then investigated the best-performing method for normalizing OOV words in situations where the training data is small and the rules for writing noisy, transliterated words using Latin letters are not limited. We enhanced the two basic seq2seq models using different beam search strategies, N-gram-based context adoption, edit distance-based correction, and dictionary-based checking in novel ways and compared their performance between each other and with the performance of two conventional methods (TM and SMT). Most of the methods we proposed improved robustness when normalizing OOV words, and all of them achieved higher word-level performance than the basic seq2seq models. Besides, most of our proposed neural methods outperformed the baseline methods such as TM and SMT when normalizing OOV words.

The next issue we discussed for the Mongolian language is to build speech synthesis system when we do not have a large amount of training data. Although recent end-to-end neural models are all able to generate natural-sounding speech, they require a large amount of training data to generate natural-sounding speech. Therefore, we proposed investigating various methods such as cross-lingual transfer learning, data augmentation, and combining the previous two methods for the low-resource Mongolian language TTS system. In other words, we showed how to train both a spectrogram prediction network and a PWG neural vocoder, which are components of

the TTS system, using only 30 minutes of Mongolian data. To determine the best performing method, we compared the performance of the TTS models using each of these methods with each other, as well as with the performance of the baseline model trained with a much larger amount (12 hours) of original target speech data. As a result, our proposed TTS system using the best performing method, a combination of cross-lingual transfer learning and data augmentation, achieved reasonable performance using only 30 minutes of original target language training data.

In addition, the methods we investigated for text normalization and speech synthesis can also be used in other low-resource languages. Although we have achieved reasonable results, the performance of our TTS system consisting of both parts needs to be further improved to apply them in a real application.

4.2 Future work

The huge increase in social media use in recent years has resulted in new forms of social interaction, changing our daily lives. Visual content on social media provides a fun and expressive way for people to communicate online. However, people with vision impairment in Mongolia are still cut off from the social media environment due to a lack of necessary tools. Therefore, people with vision impairment feel isolated and frustrated when they cannot fully participate in the interaction around visual content. In general, participation in social media is one of the major challenges for people with vision impairment. One way to address this issue at some level is to develop a TTS system, which allows people with vision impairment to connect online. It does not only help people with vision impairment, but it can also help people with hearing impairments, disabilities, aged citizens, and kids who struggle with reading. Therefore, our big goal is to build this TTS system. To this end, we will continue to study and improve the performance of the models used in the above text normalization and speech synthesis studies. Hence, we would like to increase the size of the noisy data corpora used for training and testing and try to improve the performance of our text normalization model. Furthermore, to enhance noisy text normalization, we will develop a method of identifying the languages of code-mixed sentences (e.g., sentences containing both English and transliterated

Mongolian Cyrillic text). As for speech synthesis, we will also continue investigating other TTS approaches for use with low-resource scenarios to see if we can outperform our baseline TTS model trained with a large amount of Mongolian dataset. We will then build a TTS system that can generate speech from any text, such as canonical or noisy, transliterated text, by integrating the text normalization part as a pre-processing step in the TTS system.

Besides, the preparation of public domain high-quality Mongolian speech corpus for training TTS system is very important for researchers in this field. Therefore, we will prepare high-quality, large amounts of single-speaker and multi-speaker text-speech paired data. Furthermore, we are also interested in synthesizing speech from new speakers unseen during training employing only a few seconds of speech samples. Because a deep neural network is usually trained using a corpus of several hours of recorded speech from a single speaker. The main challenge arises when it is necessary to give a new voice to a model created in this way. At this point, we need a new data corpus of this voice and to retrain the model. It will be expensive and requires great effort. Therefore, in addition to improving our previous research work, voice cloning or voice adaptation is the next task we are interested in.

References

Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray et al., "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," arXiv:1603.04467, 2016, https://www.tensorflow.org/

Shan Ariffin and Sabrina Tiun, "Rule-based Text Normalization for Malay Social Media Texts," International Journal of Advanced Computer Science and Applications, 11(10), 2020, pp.156-162.

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su, "A Phrase-based statistical model for SMS text normalization", in COLING/ACL 2006 Main Conference Poster Sessions, 17-18 July 2006, Sydney, Australia, pp. 33-40.

Kurniawati Azizah, Mirna Adriani and Wisnu Jatmiko, "Hierarchical Transfer Learning for Multilingual, Multi-Speaker, and Style Transfer DNN-Based TTS on Low-Resource Languages," IEEE access, 8, 2020, pp. 179798-179812.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," arXiv:1409.0473, 2014.

Bajibabu Bollepalli, Lauri Juvela, and Paavo Alku, "Lombard Speech Synthesis using Transfer Learning in a Tacotron Text-to-Speech System," in Interspeech 2019, 15-19 September 2019, Graz, Austria, pp. 2833–2837.

Mengnan Chen, Minchuan Chen, Shuang Liang, Jun Ma, Lei Chen, Shaojun Wang, and Jing Xiao, “Cross-Lingual, Multi-Speaker Text-To-Speech Synthesis Using Neural Speaker Embedding,” In Interspeech 2019, 15-19 September, 2019, Graz, Austria, pp. 2105-2109.

Yuan-Jui Chen, Tao Tu, Cheng-chieh Yeh, and Hung-yi Lee, “End-to-end Text-to-speech for Low-resource Languages by Cross-Lingual Transfer Learning,” in Interspeech 2019, 15-19 September 2019, Graz, Austria, pp. 2075–2079.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” in Empirical Methods in Natural Language Processing (EMNLP), 25-29 October 2014, Doha, Qatar, pp. 1724-1734.

François Chollet et al., “Keras: the Python deep-learning API,” 2015, http://keras.io/

Yu-An Chung, Yuxuan Wang, Wei-Ning Hsu, Yu Zhang, and RJ Skerry-Ryan, “Semi-supervised training for improving data efficiency in end-to-end speech synthesis,” in 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 12-17 May 2019, Brighton, UK, pp. 6940–6944.

Erica Cooper, Cheng-I Lai, Yusuke Yasuda, and Junichi Yamagishi, “Can Speaker Augmentation Improve Multi-Speaker End-to-End TTS?,” Interspeech 2020, 25-29 October, Shanghai, China, pp. 3979-3983

Munkhtulga Davaatsagaan and Kuldip K.Paliwal, “Diphone-based concatenative speech synthesis system for Mongolian”, In Proceedings of the International MultiConference of Engineers and Computer Scientists, 19-21 March 2008, Hong Kong, pp. 19-21.

Ankur Debnath, Shridevi S Patil, Gangotri Nadiger and Ramakrishnan Angarai Ganesan, “Low-Resource End-to-end Sanskrit TTS using Tacotron2, WaveGlow and Transfer Learning,” 2020 IEEE 17th India Council International Conference (INDICON), 11-13 December 2020, New Delhi, India, pp. 1-5.

Nadir Durrani, Hassan Sajjad, Hieu Hoang, and Philipp Koehn, “Integrating an unsupervised transliteration model into statistical machine translation,” in the 15th Conference of the European Chapter of the ACL (EACL 2014), 26-30 April 2014, Gothenburg, Sweden, pp. 148-153.

Mengzhe Geng, Xurong Xie, Shansong Liu, Jianwei Yu, Shoukang Hu, Xunying Liu, and Helen Meng, “Investigation of Data Augmentation Techniques for Disordered Speech Recognition,” In Interspeech 2020, 25-29 October 2020, Shanghai, China, pp. 696-700.

Sinan Göker and Burcu Can, “Neural text normalization for Turkish social media,” 2018 3rd International Conference on Computer Science and Engineering (UBMK), 20-23 September 2018, Sarajevo, Bosnia and Herzegovina, pp. 161-166.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, “On using monolingual corpora in neural machine translation”, arXiv:1503.03535, 2015.

Gerelmaa Guruuchin, “The Usages and Standard of Latin Script (In the FB frame),” The Journal of Northern Cultures Studies, 2018, 9:205-215 (in Mongolian).

Alexander Gutkin, Linne Ha, Martin Jansche, Knot Pipatsrisawat, and Richard Sproat, “TTS for Low Resource Languages: A Bangla Synthesizer,” in10th Edition of the Language Resources and Evaluation Conference, 23-28 May 2016, Portoroˇz (Slovenia), pp. 2005–2010.

Tomoki Hayashi, Ryuichi Yamamoto, Katsuki Inoue, Takenori Yoshimura, Shinji Watanabe, Tomoki Toda, Kazuya Takeda, Yu Zhang, and Xu Tan, “ESPnet-TTS: Unified, reproducible, and integratable open source end-to-end text-to-speech toolkit,” in 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 4-8 May 2020, Barcelona, pp. 7654–7658.

Lan Huang, Shunan Zhuang and Kangping Wang, “A text normalization method for speech synthesis based on local attention mechanism,” IEEE Access, 8, 2020, pp. 36202-36209

Liang Huang, Kai Zhao, and Mingbo Ma, “When to finish? Optimal beam search for neural text generation (modulo beam size)”, In 2017 Conference on Empirical Methods in Natural Language Processing, 7-11 September 2017, Copenhagen, pp. 2134–2139.

Goeric Huybrechts, Thomas Merritt, Giulia Comini, Bartek Perz, Raahil Shah, and Jaime Lorenzo-Trueba, “Low-resource expressive text-to-speech using data augmentation,” in 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 6-11 June 2021, Toronto, Ontario, Canada, pp. 6593-6597

Min-Jae Hwang, Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim, “TTS-by-TTS: TTS-driven data augmentation for fast and high-quality speech synthesis,” 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 6-11 June 2021, Toronto, Ontario, Canada, pp. 6598-6602.

Taishi Ikeda, Hiroyuki Shindo, and Yuji Matsumoto, “Japanese text normalization with encoder-decoder model,” in the 2nd Workshop on Noisy User-generated Text, 11 December 2016, Osaka, Japan, pp. 129-137.

Keith Ito, “The LJ speech dataset,” 2017, https://keithito.com/LJ-Speech-Dataset/, Accessed August 2021

Nal Kalchbrenner and Phil Blunsom, “Recurrent continuous translation models,” in Empirical Methods in Natural Language Processing (EMNLP), 18-21 October 2013, Seattle, USA, pp. 1700-1709.

Harpreet Kaur and Er. Jasdeep Singh Mann, “Text normalization using statistical machine approach,” International Research Journal of Engineering and Technology (IRJET), 8, 2016, pp. 2230-2233.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, Alexander M. Rush, “OpenNMT: Open-Source Toolkit for Neural Machine Translation”, arXiv:1701.02810, 2017.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur, “Audio Augmentation for Speech Recognition,” in Interspeech 2015, 6 - 10 September 2015, Dresden, Germany, pp. 3586-3589.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst, “Moses: Open source toolkit for statistical machine translation,” in the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions, June 2007, Prague, pp. 177-180.

Marcel de Korte, Jaebok Kim, and Esther Klabbers, “Efficient neural speech synthesis for low-resource languages through multilingual modeling,” in Interspeech 2020, 25-29 October 2020, Shanghai, China, pp. 2967–2971.

Javier Latorre, Jakub Lachowicz, Jaime Lorenzo-Trueba, Thomas Merritt, Thomas Drugman, Srikanth Ronanki, and Klimkov Viacheslav, “Effect of Data Reduction on Sequence-to-sequence Neural TTS,” in 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 12-17 May 2019, Brighton, UK, pp. 7075–7079

Younggun Lee, Suwon Shon, and Taesu Kim, “Learning pronunciation from a foreign language in speech synthesis networks,” arXiv: 1811.09364, 2020.

Vladimir I. Levenshtein, “Binary codes capable of correcting deletions,” Soviet Physics Doklady, 10 (8), 1966, pp. 707-710.

Bo Li and Heiga Zen, “Multi-Language Multi-Speaker Acoustic Modeling for LSTM-RNN based Statistical Parametric Speech Synthesis,” in Interspeech 2016, 8-12 Sep 2016, San Francisco, pp. 2468–2472.

Jingdong Li, Hui Zhang, Rui Liu, Xueliang Zhang, Feilong Bao, “End-to-End Mongolian Text-to-Speech System,” International Symposium on Chinese Spoken Language Processing (ISCSLP), 26-29 November 2018, Taipei, pp. 483-487.

Rui Liu, Feilong Bao, Guanglai Gao and Yonghe Wang, “Mongolian text-to-speech system based on deep neural network,” in National conference on man-machine speech communication, 11-13 October 2017, China, pp. 99-108.

Ruolan Liu, Xue Wen, Chunhui Lu, and Xiao Chen, “Tone Learning in Low-Resource Bilingual TTS,” in Interspeech 2020, 25-29 October 2020, Shanghai, China, pp. 2952-2956.

Ismini Lourentzou, Kabir Manghnani, and ChengXiang Zhai, “Adapting sequence to sequence models for text normalization in social media,” in the 13th International AAAI Conference on Web and Social Media, 11-14 June 2019, Münich, Germany, pp. 335-345.

Hieu-Thi Luong, Xin Wang, Junichi Yamagishi, and Nobuyuki Nishizawa, “Training Multi-Speaker Neural Text-to-Speech Systems using Speaker-Imbalanced Speech Corpora,” in Interspeech 2019, 15-19 September 2019, Graz, Austria, pp. 1303–1307

Minh-Thang Luong, Pham Hieu, and Christopher D. Manning, “Effective approaches to attention-based neural machine translation,” in Empirical Methods in Natural Language Processing (EMNLP), 17-21 September 2015, Lisbon, Portugal, pp. 1412-1421.

Massimo Lusetti, Tatyana Ruzsics, Anne Göhring, Tanja Samardžić, and Elisabeth Stark, “Encoder-Decoder methods for text normalization,” in the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018), August 2018, Santa Fe, USA, pp. 18-28.

Laurens van der Maaten and Geoffrey Hinton, “Visualizing data using t-SNE,” Journal of machine learning research, 9(86), 2008, pp. 2579–2605.

Manuel Mager, Monica Jasso Rosales, Özlema Çetinoğlu, and Ivan Meza, “Low-resource neural character-based noisy text normalization,” Journal of Intelligent & Fuzzy Systems, 36(5), 2019, pp. 4921-4929.

Alexandre Magueresse, Vincent Carles and Evan Heetderks, “Low-resource Languages: A Review of Past Work and Future Challenges”, arXiv:2006.07264, 2020.

Soumil Mandal and Karthick Nanmaran, “Normalization of transliterated words in code-mixed data using seq2seq model & Levenshtein distance,” in the 4th Workshop on Noisy User-generated Text (W-NUT), 1 November 2018, Brussels, Belgium, pp. 49-53.

NihongoDera, https://nihongodera.com/, Accessed August 2021

Yishuang Ning, Sheng He, Zhiyong Wu, Chunxiao Xing and Liang-Jie Zhang, “A Review of Deep Learning Based Speech Synthesis,” Applied Sciences, 9(19), 2019, pp. 4050.

Peter Norvig, “How to Write a Spelling Corrector (essay),” http://norvig.com/, Accessed August 2021.

Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan O. Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, and John Miller, “Deep Voice 3: 2000-speaker neural text-to-speech,” in 6th International Conference on Learning Representations (ICLR), April 30-May 3 2018, Vancouver, Canada, pp. 1094-1099.

Tatyana Ruzsics and Tanja Samardzic, “Neural sequence-to-sequence learning of internal word structure”, In 21st Conference on Computational Natural Language Learning (CoNLL’17), 3-4 August 2017, Vancouver, Canada, pp. 184–194.

Mohammad Arshi Saloot, Norisma Idris, and AiTi Aw, “Noisy text normalization using an enhanced language model,” in Artificial Intelligence and Pattern Recognition (AIPR), 17-19 November 2014, Kuala Lumpur, Malaysia, pp. 111-122.

Michael Schoeffler, Sarah Bartoschek, Fabian-Robert Stöter, Marlene Roess, Susanne Westphal, Bernd Edler, and Jürgen Herre, “webMUSHRA — A Comprehensive Framework for Web-based Listening Tests,” Journal of Open Research Software, 6(1), 2018, DOI: http://doi.org/10.5334/jors.187

Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerrv-Ryan, et al., “Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions,” in 2018 IEEE International

Conference on Acoustics, Speech and Signal Processing (ICASSP), 15-20 April 2018, Canada, pp. 4779–4783.

David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 15-20 April 2018, Canada, pp. 5329–5333.

Ryosuke Sonobe, Shinnosuke Takamichi, and Hiroshi Saruwatari, "JSUT corpus: Free large-scale Japanese speech corpus for end-to-end speech synthesis," arXiv:1711.00354, 2017, ttps://sites.google.com/site/shinnosuketakamichi/publication/just, Accessed August 2021

Jose Sotelo, Soroush Mehri, Kundan Kumar, João Felipe Santos, Kyle Kastner, Aaron Courville, and Yoshua Bengio, "Char2wav: End-to-end speech synthesis," in 5th International Conference on Learning Representations (ICLR), 24-26 April 2017, Toulon, France.

SoX: Sound eXchange audio manipulation tool, http://sox.sourceforge.net, Accessed August 2021

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le, "Sequence to sequence learning with neural networks," in Advances in Neural Information Processing Systems (NIPS), 8-13 December 2014, Montreal, Canada, pp. 3104-3112.

Hideyuki Tachibana, Katsuya Uenoyama, and Shunsuke Aihara, "Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention," in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 15-20 April 2018, Canada, pp. 4784–4788.

Noé Tits, Kevin El Haddad, and Thierry Dutoit, "Exploring Transfer Learning for Low Resource Emotional TTS," in Proceedings of SAI Intelligent Systems Conference, 5-6 September 2019, London, United Kingdom, pp. 52–60.

Osman Tursun and Ruket Cakici, “Noisy Uyghur text normalization,” in the 3rd Workshop on Noisy User-generated Text (W-NUT), 7 September 2017, Copenhagen, Denmark, pp. 85-93.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, “Attention Is All You Need,” in Advances in neural information processing systems, 2017, pp. 5998-6008.

Darnes Vilariño, David Pinto, Beatriz Beltran, Saul Leon, Esteban Castillo, and Mireya Tovar, “A machine-translation method for normalization of SMS,” in the 4th Mexican Conference on Pattern Recognition, 27-30 June 2012, Huatulco, Mexico, pp. 293-302.

Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, Rif A. Saurous, “Tacotron: Towards end-to-end speech synthesis,” in Interspeech 2017, 20-24 August 2017, Stockolm, pp. 4006-4010.

Jin Xu, Xu Tan, Yi Ren, Tao Qin, Jian Li, Sheng Zhao, Tie-Yan Liu, “LRSpeech: Extremely Low-Resource Speech Synthesis and Recognition”, In 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 23-27 August 2020, pp. 2802-2812.

Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim, “Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram,” in 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 4-8 May 2020, Barcelona, pp. 6199–6203.

Quanjie Yu, Peng Liu, Zhiyong Wu, Shiyin Kang, Helen Meng, and Lianhong Cai, “Learning cross-lingual information with multilingual BLSTM for speech synthesis of low-resource languages,” in 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 20-25 March 2016, Shanghai, China, pp. 5545–5549.

Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J Weiss, Ye Jia, Zhifeng Chen, Yonghui Wu, “LibriTTS: A corpus derived from LibriSpeech for text-to-speech,” arXiv:1904.02882, 2019

Hao Zhang, Richard Sproat, Axel H Ng, Felix Stahlberg, Xiaochang Peng, Kyle Gorman, Brian Roark, “Neural Models of Text Normalization for Speech Applications,” Computational Linguistics journal, 45(2), 2019, pp. 293-337.

Jian-dong Zhao, Guang-lai Gao, Fei-long Bao and P Mermelstein, “Research on hmm-based mongolian speech synthesis,” Computer Science journal, 41(1), 2014, pp. 80-104.

Yingbo Zhou, Caiming Xiong, and Richard Socher, “Improved regularization techniques for end-to-end speech recognition,” arXiv:1712.07108, 2017.

Appendix A

A standard for the transliteration of the Mongolian Cyrillic alphabet into the Latin alphabet (MNS 5217:2003)

#	Cyrillic letter		Latin letter		Word in Cyrillic script	Standard transliteration in Latin script
	Capital letter	Small letter	Capital letter	Small letter		
1	А	а	A	a	Ваар, аварга, аав	Vaar, avarga, aav
2	Б	б	B	b	Бага, самбар	Baga, sambar
3	В	в	V	v	Вагон, аварга, сав	Vagon, avarga, sav
4	Г	г	G	g	Газар, гэрээ, хэрэг	Gazar, geree, xereg
5	Д	д	D	d	Дадлага, ахмад	Dadlaga, axmad
6	Е	е	Ye	ye	Еэвэн	Yeeven
7	Ё	ё	Yo	yo	Ёроол	Yorool
8	Ж	ж	J	j	Жуулчин, ажил	Juulchin, ajil
9	З	з	Z	z	Зам, азагрга, бааз	Zam, azarga, baaz
10	И	и	I	i	Ишиг, бичиг	Ishig, bichig
11	-	й	-	i	Ийм, ээжийн	Iim, eejiin
12	К	к	K	k	Кино, километр	Kino, kilometer
13	Л	л	L	l	Лам, алаг, мал	Lam, alag, mal
14	М	м	M	m	Мал, хамар, нам	Mal, xamar, nam
15	Н	н	N	n	Нар, хана, үнэн	Nar, xana, u’nen
16	О	о	O	o	Орон, боловсрол, тооно	Oron, bolovsrol, toono
17	Ө	ө	O’	o’ (o)	Өдөр, өнөөдөр, шөнө	O’dor, o’noodor, sho’no
18	П	п	P	p	Пуужин	Puujin
19	Р	р	R	r	Рашаан, радио, сар	Rashaan, radio, sar
20	С	с	S	s	Сар, асар, эцэс	Sar, asar, eces
21	Т	т	T	t	Тамга, татлага	Tamga, tatlaga
22	У	у	U	u	Уран, нуруу	Uran, nuruu
23	Ү	ү	U’	u’ (u)	Үнэн, түргэн, тэргүүн	U’nen, tu’rgen, tergu’un
24	Ф	ф	F	f	Фото, фонд	Foto, fond
25	Х	х	X	x	Хавар, нөхөр, эх	Xavar, no’xor, ex
26	Ц	ц	C	c	Цацаг, цэцэг	Cacag, ceceg
27	Ч	ч	Ch	ch	Чимэг, чадал	Chimeg, chadal
28	Ш	ш	Sh	sh	Шашин, ааш	Shashin, aash
29	Щ	щ	Sch	sch	Щедрин	Schyedrin
30	-	ъ	-		Томъёо	Tomyoo
31	-	ы	-	y	Хааны, хааныг, ахын	Xaany, xaanyg, axyn
32	-	ь	-	i	Харь, барь	Xari, bari
33	Э	э	E	e	Эзэн, энэ, эмээл	Ezen, ene, emeel
34	Ю	ю	Yu	yu	Юм, юүдэн	Yum, yuu’den
35	Я	я	Ya	ya	Ямар, ядуу	Yamar, yaduu

Appendix B

A standard for the transliteration of the Mongolian Cyrillic alphabet into the Latin alphabet (MNS 5217:2012)

#	Cyrillic letter		Latin letter		Word in Cyrillic script	Standard transliteration in Latin script
	Capital letter	Small letter	Capital letter	Small letter		
1	А	а	A	a	Аварга, халбага, аав	Avarga, khalbaga, aav
2	Б	б	B	b	Бага, самбар	Baga, sambar
3	В	в	V	v	Вагон, аварга, сав	Vagon, avarga, sav
4	Г	г	G	g	Газар, гэрээ, хэрэг	Gazar, geree, khereg
5	Д	д	D	d	Дадлага, ахмад	Dadlaga, akhmad
6	Е	е	Ye	ye	Еэвэн, ерөөл	Yeeven, yerööl
7	Ё	ё	Yo	yo	Ёроол, оёдол	Yorool, oyodol
8	Ж	ж	J	j	Жуулчин, ажил	Juulchin, ajil
9	З	з	Z	z	Зам, заавар	Zam, zaavar
10	И	и	I	i	Ишиг, бичиг, хань	Ishig, bichig, khani
11	-	й	-	i	Ийм, ээжийн	Iim, eejiin
12	К	к	K	k	Кино, километр	Kino, kilometer
13	Л	л	L	l	Лам, алаг, мал	Lam, alag, mal
14	М	м	M	m	Мал, хамар, нам	Mal, xamar, nam
15	Н	н	N	n	Нар, хана, үнэн	Nar, khana, ünen
16	О	о	O	o	Орон, боловсрол, тооно	Oron, bolovsrol, toono
17	Ө	ө	Ö	ö	Өдөр, өнөөдөр, өөрөөсөө	Ödör, önöödör, ööröösöö
18	П	п	P	p	Пуужин	Puujin
19	Р	р	R	r	Рашаан, радио, сар	Rashaan, radio, sar
20	С	с	S	s	Сар, асар, эцэс	Sar, asar, etses
21	Т	т	T	t	Тамга, татлага	Tamga, tatlaga
22	У	у	U	u	Уран, нуруу	Uran, nuruu
23	Ү	ү	Ü	ü	Үнэн, түргэн, тэргүүн	Ünen, türgen, tergüün
24	Ф	ф	F	f	Фото, фонд	Foto, fond
25	Х	х	Kh	kh	Хавар, нөхөр, эх	Khavar, nökhör, ekh
26	Ц	ц	Ts	ts	Цацаг, цэцэг	Tsatsag, tsetseg
27	Ч	ч	Ch	ch	Чимэг, чадал, ач	Chimeg, chadal, ach
28	Ш	ш	Sh	sh	Шашин, ааш	Shashin, aash
29	Щ	щ	Sh	sh	Щедрин	Shyedrin
30	-	ъ	-	i	Оръё, суръя, гаръя	Oriyo, suriya, gariya
31	-	ы	-	y	Хааны, ахын	Khaany, akhyn
32	-	ь	-	i	Харь, барь	Khari, bari
33	Э	э	E	e	Эзэн, энэ, эмээл	Ezen, ene, emeel
34	Ю	ю	Yu	yu	Юм, юүдэн	Yum, yuüden
35	Я	я	Ya	ya	Ямар, ядуу, ая	Yamar, yaduu, aya