BTS: Back TranScription for Speech-to-Text Post-Processor using Text-to-Speech-to-Text

Chanjun Park[1], **Jaehyung Seo**[1], **Seolhwa Lee**[1], **Chanhee Lee**[1]
Hyeonseok Moon[1], **Sugyeong Eo**[1], **Heuiseok Lim**[1†]
[1]Korea University, South Korea
{bcj1210,seojae777,whiteldark, chanhee0222}@korea.ac.kr
{glee889, djtnrud, limhseok}@korea.ac.kr

Abstract

With the growing popularity of smart speakers, such as Amazon Alexa, speech is becoming one of the most important modes of human-computer interaction. Automatic speech recognition (ASR) is arguably the most critical component of such systems, as errors in speech recognition propagate to the downstream components and drastically degrade the user experience. A simple and effective way to improve the speech recognition accuracy is to apply automatic post-processor to the recognition result. However, training a post-processor requires parallel corpora created by human annotators, which are expensive and not scalable. To alleviate this problem, we propose Back TranScription (BTS), a denoising-based method that can create such corpora without human labor. Using a raw corpus, BTS corrupts the text using Text-to-Speech (TTS) and Speech-to-Text (STT) systems. Then, a post-processing model can be trained to reconstruct the original text given the corrupted input. Quantitative and qualitative evaluations show that a post-processor trained using our approach is highly effective in fixing non-trivial speech recognition errors such as mishandling foreign words. We present the generated parallel corpus and post-processing platform to make our results publicly available.

1 Introduction

Automatic speech recognition (ASR) is a technology that converts human voice into text. With the emergence of deep learning, the performance of ASR has been improved considerably. Consequently, many firms are applying ASR to their business models (Kaya et al., 2020).

Although several excellent commercial API systems are available, such as Google Cloud Speech API (Aleksic et al., 2015) and Naver's CLOVA Speech (Chung, 2019), most small- and medium-sized companies are building their own ASR software using open-source tools such as Kaldi (Povey et al., 2011) owing to the need for domain-specific systems as well as security of in-house industrial data (Vajpai and Bora, 2016). In addition, many companies are operating on conventional ASR architectures, such as Gaussian mixture models (GMMs) (Stuttle, 2003) and hidden Markov models (HMMs) (Gales and Young, 2008), which are based on acoustic and language models.

However, a drawback of the above-mentioned method is that words that are not in the dictionary are misrecognized as incorrect words owing to the out-of-vocabulary (OOV) problem. As this method is a statistics-based method, satisfactory performance is achieved only when a massive voice database is available . Probability values for sequences of words that are not present in the training corpus are estimated to be unstable, and it is difficult to sufficiently reflect the context because n values are constrained in n-grams. Moreover, the entry barrier is high because it is difficult for non-professionals to handle the model.

To alleviate these limitations, ASR studies have recently been conducted using pretrained model (PM)-based transfer learning (Baevski et al., 2020; Hjortnæs et al., 2021; Zhang et al., 2021). This methodology shows superior performance compared to methods based on the conventional ASR architecture; however, it has two main limitations in terms of applying it to real-world services.

First, from the data aspect, this methodology requires a large amount of training data for pretraining to service the ASR software. As it is strongly dependent on the data size, it is difficult to apply it to a low-resource language (LRL), such as Korean. Furthermore, as the latest studies are based on a high-resource language (HRL) with sufficient training data, the same performance cannot be achieved

† Corresponding author

Proceedings of the 8th Workshop on Asian Translation, pages 106–116
Bangkok, Thailand (online), August 5-6, 2021.

if the same model is applied to an LRL without any special processing.
Second, from the service environment aspect, this methodology requires service circumstances with sufficient computing power (e.g., GPU) to process large-scale data. It is difficult to establish a sufficient hardware environment to provide services, except for large companies such as Google and Facebook. In other words, as training a model involves many parameters and a large amount of data, companies that do not have sufficient server or GPU environments will find it difficult to configure the service environment and improve performance using the latest model (Park et al., 2020c). Therefore, it is important to ensure that companies with insufficient environments can provide services while performing well against LRLs. To this end, instead of PM-based transfer learning, a new method for improving ASR performance is required.

To alleviate these limitations, some studies have attempted to improve the performance of the ASR model through various pre-processing and post-processing methods without changing the model (Jeong et al., 2003; Jung et al., 2004; Voll et al., 2008; Mani et al., 2020; Liao et al., 2020). This approach does not require a large amount of data for pretraining the model and it can be applied to any model as well as models that can provide sufficient service with a CPU (Klein et al., 2020), such as the vanilla Transformer (Vaswani et al., 2017). In this regard, this method can alleviate the above-mentioned limitations in terms of the data and service environment. Hence, this method is particularly important from the viewpoint of LRLs.

Accordingly, we propose Back TranScription (BTS), a fully automated data construction method for a sequence-to-sequence (S2S)-based post-processor model that does not require human intervention or model modification. The contributions of this study are as follows.

- We propose BTS, a simple and effective method for generating ASR post-processor training corpus without expensive human labor. As this approach does not require human intervention, it can create a vast amount of training data from raw text, which drastically reduces the cost of building such a model.
- We discuss the characteristics and effectiveness of our approach on the basis of extensive quantitative and qualitative evaluations.
- We present the generated parallel data and post-processing platform to make our results publicly available*.

2 Related Work

ASR post-processing is a research field that aims to improve performance by correcting the ASR errors rather than changing the model architecture. The two main methodologies for ASR post-processing in the field of speech recognition are the conventional methodology and the sequence-to-sequence (S2S) methodology.

Conventional Methodology The conventional methodology is based on rules and statistics. Firms attempt to improve ASR performance by building their own rules while providing ASR services. They apply linguistic rules to improve the quality of the speech recognition results. The drawback of this methodology is that it involves high costs and requires a long time to produce abundant rules. Moreover, conflicts between rules may occur. Furthermore, each component must be implemented independently (Paulik et al., 2008; Škodová et al., 2012). Some post-processing studies have been conducted using the N-gram language model; however, the statistics-based method requires a large amount data and cannot consider the context (Cucu et al., 2013; Bassil and Semaan, 2012).

Sequence-to-Sequence (S2S) Methodology The S2S methodology corrects errors in the same way as the machine translation process (Vaswani et al., 2017; Baskar et al., 2019; Park et al., 2020a). Based on the S2S model, the STT result is vectorized using an encoder and the vector is then decoded to generate a human-modified STT sentence. This methodology outperforms the conventional method based on rules and statistics. However, the ASR post-processor based on the S2S methodology has some limitations in terms of data construction and industrial service.

First, from the data construction aspect, no open data are available for training, and a parallel corpus must be manually built for the ASR post-processor. The training data are of the form (speech recognition sentence, human post-edit sentence), and constructing such data involves human intervention to transcribe the speech. In other words, considerable time and effort are required to construct the data. In addition, quality differences may occur depending

*http://nlplab.iptime.org:32260/

on the transcriber. Different individuals may transcribe the same sentence differently, resulting in performance degradation of the model. Hence, we aim to alleviate the limitations of S2S-based data construction through BTS using Text-to-Speech-to-Text (TST). This method can reduce the cost and time required for data construction and is free of the quality issues related to human transcription.

Second, from the service aspect, although most recent NLP studies are based on the pretrain-finetuning approach (PFA), small- and medium-sized enterprises lack sufficient hardware; hence, there are many limitations in terms of using the technology to service NLP application software owing to low speed and insufficient memory. Although methods such as XLM (Lample and Conneau, 2019), MASS (Song et al., 2019), and mBART (Liu et al., 2020) show the best current performance, the corresponding models are too large in terms of the number of parameters and model size. Therefore, it is still unreasonable to provide practical services in the industry. Furthermore, as this methodology is dependent on the data size, it can be easily applied to an HRL whereas its application to an LRL is limited.

This study is similar to studies on automatic post-editing (APE) (Chatterjee et al., 2019) and grammar error correction (GEC) (Bryant et al., 2019). However, APE performs post-processing on machine translation results while GEC is designed to correct grammar, i.e., their post-processing targets are different. In addition, these methods are mainly based on an HRL-based pretrained language model (PLM) such as XLM, MASS, or UniLM (Dong et al., 2019). Hence, it is difficult to apply them to services provided by small- and medium-sized enterprises with insufficient environments.

In this study, we use the vanilla Transformer, which can be easily applied to the required service. In contrast to previous studies, we conduct an experiment on the Korean language, which is an LRL, and we make the model constructed in this study freely available.

3 Proposed Method

3.1 Background

In this study, we introduce four mainstream attributes reflecting the readability and satisfactoriness of ASR service in order to provide high-quality service to end users of our BTS mechanism, which can be used for training the ASR post-processor.

Spacing The first limitation is related to segmentation, i.e., the spaces are generally not adequately separated in the speech recognition result. To solve this problem, many studies have investigated an automatic spacing module; however, few studies have focused on ASR (Lee and Kim, 2013; Choi et al., 2021). Thus, the satisfactoriness of ASR service, which is used by end users, is low, and the speech recognition results will lack credibility if this problem is not resolved.

Foreign Word Conversion The second limitation is the foreign word conversion problem. For example, for the sentence "`The Lotte tower is on the 123rd floor.`", the ASR service outputs "`The` 롯데 타워 `is on the 123rd floor.`". In other words, 롯데 타워 is not converted into `Lotte tower`. We refer to this problem as the foreign word conversion problem. Although it is not a critical problem, solving it can improve the readability and satisfactoriness of the ASR system for end users.

Punctuation The third limitation is related to punctuation (e.g., period, comma, exclamation and question marks). The correct output of the ASR system should be "`where are you going?`"; however, the general ASR system outputs "`where are you going`" without the question mark. Thus, the lack of punctuation makes it problematic for the end users to understand the purpose of sentence segmentation. This could lead to complex issues in the recognition of end users' utterance intentions in terms of who wants to use the output. Furthermore, commercial ASR systems typically do not use punctuation when they provide services (Ha et al., 2020). Several studies (Yi et al., 2020; Guan, 2020) have attempted to solve these punctuation problems independently.

Spelling Errors The fourth limitation is related to spelling errors, which frequently occur in the ASR result. Although previous studies (Kiyono et al., 2019; Choe et al., 2019; Park et al., 2020a) have investigated spelling correction, few have focused on ASR.

3.2 Back TranScription (BTS)

BTS is a technique that is integrated with TTS and STT to yield a parallel corpus. The process of building a parallel corpus involves the following steps:

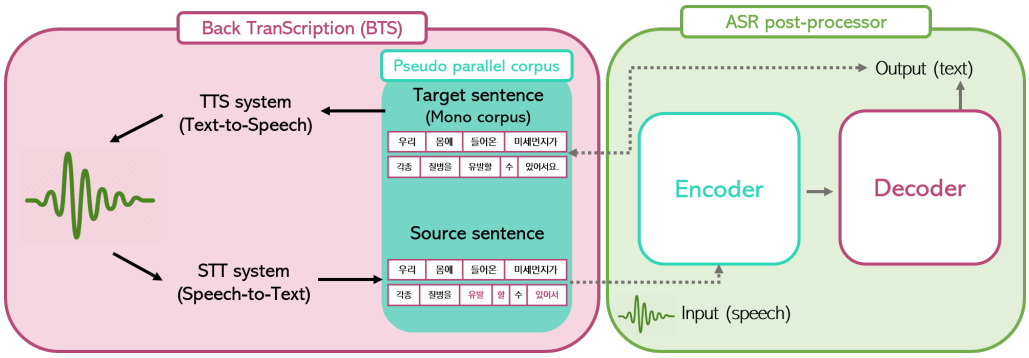

Figure 1: Overall architecture of BTS and ASR post-processor. Note that the red words in the source sentence are ungrammatical words. The source sentence means "Fine dust occurs many diseases when it comes to our bo" and the target sentence means "Fine dust occurs many diseases when it comes to our body.".

	No Filter				Filter				Test	
	Train		Valid		Train		Valid			
	src	**tgt**	**src**	**tgt**	**src**	**tgt**	**src**	**tgt**	**src**	**tgt**
# of sents	224,987	224,987	5,000	5,000	214,318	214,318	5,000	5,000	5,000	5,000
# of tokens	7,319,788	7,731,924	160,773	167,888	7,117,361	7,447,350	136,496	143,044	165,781	172,590
# of words	1,950,669	1,900,409	42,968	41,780	1,887,843	1,830,500	36,655	35,961	43,482	41,472
avg of SL △	32.53	34.37	32.15	33.58	33.21	34.75	27.3	28.61	33.16	34.52
avg of WS	8.67	8.45	8.59	8.36	8.81	8.54	7.33	7.19	8.7	8.29
avg of SS	7.67	7.45	7.59	7.36	7.81	7.54	6.33	6.19	7.7	7.29
# of K-toks ∗	5,503,227	5,599,442	121,131	122,502	5,365,092	5,420,594	103,667	104,961	123,566	124,203
# of E-toks	32,389	61,294	504	829	30,217	57,203	463	724	1,162	2,262
# of S-toks	2,181	338,459	55	6,675	1,769	307,339	21	5,682	74	6,873

Table 1: Statistics of our parallel corpus results on TST with and without a filter. We define the original colloquial sentences as the target (tgt) and the generated sentences after TST as the source (src). Moreover, we attempt to identify the linguistic features of our parallel corpus, including # of sents/tokens/words (number of sentences/tokens/words); △ avg of SL/WS/SS (average of sentence length/words/spaces per sentence); and ∗ # of K/E/S-toks (number of Korean/English/special-symbol letter tokens).

1) crawling the pre-built mono corpus in a convenient manner; 2) transformation into speech using TTS; and 3) outputting the converted result as text using STT. We aim to apply the BTS mechanism to the mainstream attributes described in Section 3.1.

In other words, we apply TST to the result of TTS (i.e., the original mono corpus) and then create a pseudo-parallel corpus for the ASR post-processor. This can be explained in terms of machine translation as follows: the source sentence is substituted for the output of TST, and the original mono corpus is substituted for the target sentence.

TST is the processor for creating ASR errors from the mono corpus, i.e., the ASR errors can be generated through TST, which is integrated with both the TTS module that provides the data for the STT module and the STT module. The mono corpus is grounded well in space, can handle foreign word conversion and punctuation, and rarely involves spelling errors. We can develop a high-quality ASR post-processor based on S2S training using these processes.

Figure 1 shows the proposed method, including the BTS architecture and ASR post-processor architecture based on S2S training using the pseudo-parallel corpus, which is derived from BTS. The module on the left (BTS) shows the target sentence (ground sentence) converted into speech using the TTS system, which is then converted into the source sentence (error sentence) through the STT system. The module on the right (ASR post-processor) shows the S2S-based ASR post-processor, which uses the speech of the source sentence as the model input and the target sentence as the ground truth. In the BTS module, the pseudo-parallel corpus consists of the target sentence from the mono corpus and the source sentence converted from the TTS output (i.e., speech) into the STT output (i.e., text). The source sentence includes the above-mentioned errors. Finally, we can train

the ASR post-processor using the pseudo-parallel corpus.

3.3 Why BTS?

The advantages of the BTS method in terms of service can be attributed to five factors.

- First, BTS can build infinite training data for ASR or other purposes. In general, building a parallel corpus is expensive and time-consuming. Moreover, it is difficult to establish a high-quality parallel corpus. However, we can easily build an infinite parallel corpus if we exploit the advantages of the mono corpus through web crawling.

- Second, BTS supports a universal method for integrating solutions to problems such as spacing, foreign word conversion, punctuation, and spelling errors using a single model, as the mono corpus that is used in our method is free of the above-mentioned problems. Previous studies have been conducted independently, whereas our method can resolve these issues simultaneously.

- Third, commercial ASR systems such as Google Cloud Speech API can be converted into domain-specific ASR systems. If a TTS is produced using only a single corpus of the specific domain and a post-processor is created using the constructed parallel corpus, the commercial ASR system can be serviced with a domain-specific ASR. Companies build their own ASR system rather than using commercial systems because of the need for a domain-specific model, which can be built by exploiting the high recognition rate of a commercialized system through BTS. We define these domain corrections.

- Fourth, our method does not require human intervention for building a parallel corpus as it involves automatic generation; therefore, it achieves significant time and cost savings. In addition, it is free of the quality issues that may arise in the case of different human operators.

- Finally, language extension is simple and convenient. The commercial system (Aleksic et al., 2015) provides various TTS and STT language-specific API services. Therefore, we can collect a diverse language dataset for BTS.

In summary, BTS is a practical solution that can enable companies to provide ASR service.

4 Experimental Setup

4.1 Data Collection

Build Mono Corpus The parallel corpus for experimenting with BTS was set to Korean, which is an LRL, and we collected it from two different sources. First, we extracted 129,987 sentences from the business and technology TED provided in a script translated into Korean. Second, we extracted 105,000 sentences from the Korean-English translation corpus in AI-HUB (Park and Lim, 2020)

TTS Using the mono corpus, we converted the text into voice data in the mp3 format using Google TTS API. Specifically, 129,987 sentences from TED were divided into 7,969,230 speech tokens and synthesized with 2,081,115 s of voice data. Further, 105,000 sentences from AI-HUB were divided into 3,065,086 speech tokens and synthesized with 1,563,990 s of voice data. The voice data were synthesized using the same WaveNet model (Oord et al., 2016) as that used for Google Assistant, Google Search, and Google Translation, which required less than 36 h and 24 h for the conversion, respectively. The commercialized API system was used to lower the entry barrier, thereby allowing companies that lack a TTS system to use BTS.

STT The voice data constructed by TTS use Navers CLOVA Speech Recognition (CSR) API to proceed with the conversion back to text data. The speech recognition API uses the same model as that used for Navers Voice Recognition Notes and Searches, which requires less than 120 h and 72 h for the conversion, respectively. After this process, a parallel corpus of 229,987 sentence pairs, consisting of the target sentences prior to speech synthesis and recognition as well as the translated source sentences, is built for the S2S-based ASR post-processors.

Parallel Corpus Filtering Parallel corpus filtering (PCF) (Koehn et al., 2020) is the process of constructing a qualitatively validated parallel corpus. In other words, it is a sub-field of machine translation in which training data are selected to ensure high-quality training to improve the performance of the model.

In the case of the pseudo-parallel corpus built through TST, some source sentences are empty or

Model	BLEU	GLEU
Base	42.19	N/A
Park et al. (2020a)	50.62 (+8.43)	31.79
No-Filter	55.72 (+13.53)	46.23
Filter	**56.56 (+14.37)**	**46.94**

Table 2: Overall BTS performance verification results

too short; they are not recognized owing to unintentional errors in the STT and TTS systems. Thus, we use the PCF methodology proposed by Park et al. (2020b) to obtain only high-quality data. A total of 10,669 sentences are filtered, most of which are low-quality data obtained because of poor recognition during STT. In addition, we remove pairs of sentences that are identical or which consist of special symbol tokens comprising more than 50% of the total tokens , as these sentences may not be inconsistent with the learning method.

Final Constructed Pseudo-Parallel Corpus We compared the performance of the PCF-driven model with that of the non-progress model to verify the effectiveness of filtering. For models without filtering (No-Filter), the training data included 224,987 sentences and the verification data included 5,000 sentences. For the filtered (Filter) model, the training data included 214,318 sentences and the verification data included 5,000 sentences. In the case of the test set, 5,000 sentences of the No-Filter version were constructed to evaluate the performance changes depending on whether filtering was applied during the training process.

4.2 Model

For the post-processor, we trained the vanilla Transformer with the pseudo-parallel corpus, generated by BTS. The hyper-parameter settings were the same as those used by Vaswani et al. (2017). Further, we used SentencePiece (Kudo and Richardson, 2018) for sub-word tokenization and set the vocabulary size to 32,000. Two GTX 1080ti GPUs were used in the experiments.

5 Experimental Results

5.1 Data statistics and analysis

Using BTS, we constructed a parallel corpus for an S2S-based ASR post-processor with 219,318 sentences that are finally processed by PCF.

We conducted a statistical analysis of the constructed corpus and a comparative analysis with and without PCF. The results are summarized in Table 1.

First, we conducted basic analyses, such as the number of data, number of tokens, and average length of sentences. The lengths of the source sentences built using BTS, regardless of whether the filter was applied, were smaller than those of the target sentences on average 1.69, 1.37, and 1.36 for the training, validation, and test datasets, respectively. However, the average numbers of source sentence words were greater than those of target sentence words on average 0.245, 0.185, and 0.41 for the training, validation, and test datasets, respectively. Considering the average number of blank spaces, these results are attributed to the unnecessary separation of phrases, even though the source sentences have a relatively small total number of tokens.

Second, we analyzed the Korean and English tokens. In the case of K-tokens, 75,859, 1,333, and 672 tokens in the training, validation, and test datasets were lost in the source sentences, respectively. Token loss is the reason for the omission of sentence endings and suffixes, and it is estimated that the model reflects the common characteristics of Korean speakers who pronounce the ending in a slurred manner. In addition, the E-tokens are transformed into Korean tokens as pronounced and suitable phonetic values are not obtained. Consequently, 27,946, 293, and 1,100 E-tokens in the training, validation, and test dataset were lost in the source sentences, respectively.

Third, the most significant loss was in the case of S-tokens. The source sentences in the training, validation, and test datasets lost 320,924, 6,141, and 6,799 special symbol tokens, respectively. For example, periods, commas, exclamation marks, and small brackets, which are added to describe the situation in the transcription of the original data, tend to be lost in the source sentences. Such special symbol tokens may sometimes contain actual colloquial tones or emotions that are not represented by the text adequately. Thus, excessive omission of special symbol tokens is equivalent to the loss of rich representation information of colloquial forms that are different from written ones.

In conclusion, we make the model that we have constructed freely available in order to lower the entry barrier for research institutions and mitigate the cost challenges faced by many small- and medium-sized enterprises that lack sufficient resources.

Model	Spacing	Word Conversion(KO)	Word Conversion(EN)	Punctuation	Overall
Base	89.60	46.87	4.14	0.89	61.84
Filter	**91.86 (+2.26)**	**54.41 (+7.54)**	**23.41 (+19.27)**	**61.02 (+60.13)**	**70.73 (+8.89)**

Table 3: BTS performance classified as Automatic Spacing, Word Conversion, Punctuation, and Overall. The F1 scores are reported for each feature. KO: Korean; EN: English.

Input Sentence	Output Sentence	Type of Error
제가 어렸을 때 우연히 **티비**에서 박지성 선수가 축구 경기를 하는 모습을 보았어요 (When I was young, I watched Park Ji-sung playing a soccer on **티비**.)	제가 어렸을 때 우연히 **TV**에서 박지성 선수가 축구 경기를 하는 모습을 보았어요. (When I was young, I watched Park Ji-sung playing a soccer on **TV**.)	Foreign Word Conversion
참석자 명단을 받으려면 어느 부서에 가야 하나 (Which department should I go to receive a list of attendees)	참석자 명단을 받으려면 어느 부서에 가야 하나**?** (Which department should I go to receive a list of attendees**?**)	Punctuation
너무 보고 싶은데 **볼수가** 없네 (I **missyou** so much, but I can't)	너무 보고 싶은데 **볼 수가** 없네. (I **miss you** so much, but I can't.)	Spacing
가평은 한국의 유명한 관광지 (Gapyeong a famous tourist spot in Korea)	가평은 한국의 유명한 관광지**에요.** (Gapyeong **is** a famous tourist spot in Korea.)	Spelling errors
감자탕은 무료 서비스로 **나온** (Gamjatang is **serv** for free)	감자탕은 무료 서비스로 **나옵니다.** (Gamjatang is **served** for free.)	Spelling errors

Table 4: Examples of sentences for qualitative analysis

5.2 Quantitative Results

Spelling Correction First, we verified the spelling correction performance of the BTS-based ASR post-processor. We employed GLEU (Napoles et al., 2015) and BLEU (Papineni et al., 2002) as the performance evaluation metrics. GLEU is similar to BLEU; the difference is that GLEU also considers the source information and is specialized for spelling error correction systems.

Base refers to the BLEU score between the source and target sentences; we leveraged it as the baseline for assessing the performance improvement. In addition, we compared the performance with that of the Korean spelling error correction model proposed by Park et al. (2020a), who performed ASR post-processing experiments and published the model as a demo system[†]. This study focused on Korean spelling error correction that is not specialized in ASR post-processing. However, as the experiments were performed with respect to speech recognition error correction, we compared the performance of this model with that of the proposed model. Through this comparison, we could assess the spelling correction performance of our approach. The experimental results are summarized in Table 2.

Our results show that PCF can improve the correction performance. The BLUE and GLEU scores of the No-Filter model were 55.72 and 46.23, respectively. The BLEU score was higher than that of the base model by 13.53. Further, the BLEU and GLEU scores of the Filter model were 56.56 and 46.94, respectively. The BLEU score was higher than that of the base model by 14.37. Thus, PCF can promote performance improvement.

Furthermore, the BLEU and GLEU scores of the Filter model were higher than those of the existing spelling correction model proposed by Park et al. (2020a) by 5.94 and 15.15, respectively. These results show that our post-processor can achieve higher performance in spelling correction.

Automatic Spacing Second, we verified the performance of the BTS-based post-processor in automatic spacing. To measure the multi-class accuracy, we used the F1-score to correctly locate the spacing in the target sentences. As the Filter model achieves better performance (see Table 2), further experiments were based on the Filter model. Our results can be found in the Spacing part of Table 3.

Using the post-processor, we achieved a scored that was higher than that of the base model by 2.26. Thus, BTS can promote correct automatic spacing.

Foreign Word Conversion Third, we demonstrated the performance of the BTS-based post-processor in foreign word conversion. For the performance evaluation, we used the F1-score to correctly locate Korean and English words in the target sentences. The experimental results are shown in

[†] http://nlplab.iptime.org:32288/

the Word Conversion part of Table3.

Compared to the base model, our processor yielded scores that were higher by 7.54 and 19.27 for Korean and English word conversion, respectively. From these results, we can conclude that our post-processor facilitates better performance in word conversion.

Punctuation Attachment Finally, we verified the performance of the BTS-based post-processor in punctuation attachment. For the performance evaluation, we used the F1-score to correctly locate the punctuation in the target sentences. Our results are presented in the Punctuation part of Table 3.

The performance score of the BTS-based post-processor was 60.13 higher than that of the base model. For the base model, the F1-score of punctuation attachment was 0.89, which indicates that the base model rarely achieves correct punctuation attachment. This represents the limitation of commercial STT systems. For the test set, the base model only attached the period (“.”) 32 times, the percent sign (“%”) 33 times, and the dollar symbol (“$”) 1 time. These limitations can be alleviated by applying our method. The BTS-based post-processor facilitates the attachment of the above-mentioned punctuation marks as well as other fundamental punctuation marks, such as the question mark(“?”), exclamation mark(“!”), and comma(“,”), in colloquial sentences. Thus, our proposed post-processor can achieve tremendous improvement in punctuation attachment.

Thus, we have shown the performance improvement for the four above-mentioned criteria. Furthermore, the overall F1-score, calculated by considering all these criteria in one step, showed an improvement of 8.89.

5.3 Qualitative Analysis

In addition to the quantitative analysis described above, we also performed qualitative analysis. Table 4 lists some examples of source sentences and the corrected output of each sentence, generated by the BTS-based ASR post-processor. As shown in Table 4, the BTS-based ASR post-processor can effectively correct errors arising in ASR models.

First, the post-processor can correct foreign word conversion errors. In Korean sentences, the foreign word “TV” is generally adopted in its original form; however, the ASR system transcribes this word with its Korean pronunciation, “티비”. Our results show that the BTS-based post-processor can effectively correct this error.

Second, it is possible to correct punctuation attachment errors and inappropriate spacing, which frequently occur in the ASR model. A period (“.”) or question mark(“?”) can be correctly attached to each sentence, and spacing errors such as “missyou” can be corrected as ”miss you”. Through this revision process, we expect that end users can be provided with clearer sentences.

Third, the BTS-based ASR post-processor can correct spelling errors or improper sentence endings generated by the speech recognition system. In particular, for Korean, improper sentence endings often lead to different interpretations of whole sentences. These issues can be effectively rectified by our method.

For example, in the case of a missing sentence ending, the post-processor can restore the sentence by attaching the omitted part “에요(is)”. In addition, the word “serv”, which occurs owing to the recognition error of the sentence ending, can be corrected with the appropriate word “나옵니다(served)”.

In summary, by inspecting examples of BTS-based post-processing, we can conclude that BTS is an effective approach for dealing with spelling correction, automatic spacing, foreign word conversion, and punctuation attachment. This study is significant in that errors of ASR systems can be corrected without human-labeled data, which require professional human resources for generation.

6 Conclusion and Future Work

We proposed BTS, which can automatically generate a parallel corpus from raw corpora to train ASR post-processors. By combining TTS and STT systems, ASR noise was injected into the raw text, and the post-processing model was trained in a denoising manner. Quantitative and qualitative evaluations showed that our approach can effectively handle challenging ASR errors, such as foreign word conversion.

In the future, we plan to investigate different noising strategies that reflect real-world ASR errors and make the denoising process more challenging. Demonstrating the effectiveness of BTS in additional languages from various language families is another important direction for future research.

Acknowledgments

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2018-0-01405) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation), Institute for Information & communications Technology Planning & Evaluation (IITP), grant funded by the Korean government (MSIT) (No. 2020-0-00368, A Neural-Symbolic Model for Knowledge Acquisition and Inference Techniques) and MSIT(Ministry of Science and ICT), Korea, under the ICT Creative Consilience program(IITP-2021-2020-0-01819) supervised by the IITP(Institute for Information & communications Technology Planning Evaluation).

References

Petar Aleksic, Mohammadreza Ghodsi, Assaf Michaely, Cyril Allauzen, Keith Hall, Brian Roark, David Rybach, and Pedro Moreno. 2015. Bringing contextual information to google speech recognition.

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *arXiv preprint arXiv:2006.11477*.

Murali Karthick Baskar, Shinji Watanabe, Ramon Astudillo, Takaaki Hori, Lukáš Burget, and Jan Černockỳ. 2019. Semi-supervised sequence-to-sequence asr using unpaired speech and text. *arXiv preprint arXiv:1905.01152*.

Youssef Bassil and Paul Semaan. 2012. Asr context-sensitive error correction based on microsoft n-gram dataset. *arXiv preprint arXiv:1203.5262*.

Christopher Bryant, Mariano Felice, Øistein E Andersen, and Ted Briscoe. 2019. The bea-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75.

Rajen Chatterjee, Christian Federmann, Matteo Negri, and Marco Turchi. 2019. Findings of the wmt 2019 shared task on automatic post-editing. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 11–28.

Yo Joong Choe, Jiyeon Ham, Kyubyong Park, and Yeoil Yoon. 2019. A neural grammatical error correction system built on better pre-training and sequential transfer learning. *arXiv preprint arXiv:1907.01256*.

Jeong-Myeong Choi, Jong-Dae Kim, Chan-Young Park, and Yu-Seop Kim. 2021. Automatic word spacing of korean using syllable and morpheme. *Applied Sciences*, 11(2):626.

Joon Son Chung. 2019. Naver at activitynet challenge 2019–task b active speaker detection (ava). *arXiv preprint arXiv:1906.10555*.

Horia Cucu, Andi Buzo, Laurent Besacier, and Corneliu Burileanu. 2013. Statistical error correction methods for domain-specific asr systems. In *International Conference on Statistical Language and Speech Processing*, pages 83–92. Springer.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*.

Mark Gales and Steve Young. 2008. The application of hidden markov models in speech recognition.

Yushi Guan. 2020. End to end asr system with automatic punctuation insertion. *arXiv preprint arXiv:2012.02012*.

Jung-Woo Ha, Kihyun Nam, Jin Gu Kang, Sang-Woo Lee, Sohee Yang, Hyunhoon Jung, Eunmi Kim, Hyeji Kim, Soojin Kim, Hyun Ah Kim, et al. 2020. Clovacall: Korean goal-oriented dialog speech corpus for automatic speech recognition of contact centers. *arXiv preprint arXiv:2004.09367*.

Nils Hjortnæs, Niko Partanen, Michael Rießler, and Francis M Tyers. 2021. The relevance of the source language in transfer learning for asr. In *Proceedings of the Workshop on Computational Methods for Endangered Languages*, volume 1, pages 63–69.

Minwoo Jeong, Byeongchang Kim, and Gary Geunbae Lee. 2003. Semantic-oriented error correction for spoken query processing. In *2003 IEEE Workshop on Automatic Speech Recognition and Understanding (IEEE Cat. No. 03EX721)*, pages 156–161. IEEE.

Sangkeun Jung, Minwoo Jeong, and Gary Geunbae Lee. 2004. Speech recognition error correction using maximum entropy language model. In *Eighth International Conference on Spoken Language Processing*.

Sema Kayapinar Kaya, Turan Paksoy, and Jose Arturo Garza-Reyes. 2020. The new challenge of industry 4.0. *Logistics 4.0: Digital Transformation of Supply Chain Management*, page 51.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. *arXiv preprint arXiv:1909.00502*.

Guillaume Klein, Dakun Zhang, Clément Chouteau, Josep M Crego, and Jean Senellart. 2020. Efficient and high-quality neural machine translation with opennmt. In *Proceedings of the Fourth Workshop on Neural Generation and Translation*, pages 211–217.

Philipp Koehn, Vishrav Chaudhary, Ahmed El-Kishky, Naman Goyal, Peng-Jen Chen, and Francisco Guzmán. 2020. Findings of the wmt 2020 shared task on parallel corpus filtering and alignment. In *Proceedings of the Fifth Conference on Machine Translation*, pages 726–742.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Changki Lee and Hyunki Kim. 2013. Automatic korean word spacing using pegasos algorithm. *Information processing & management*, 49(1):370–379.

Junwei Liao, Sefik Emre Eskimez, Liyang Lu, Yu Shi, Ming Gong, Linjun Shou, Hong Qu, and Michael Zeng. 2020. Improving readability for automatic speech recognition transcription. *arXiv preprint arXiv:2004.04438*.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Anirudh Mani, Shruti Palaskar, Nimshi Venkat Meripo, Sandeep Konam, and Florian Metze. 2020. Asr error correction and domain adaptation using machine translation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6344–6348. IEEE.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Chanjun Park, Kuekyeng Kim, YeongWook Yang, Minho Kang, and Heuiseok Lim. 2020a. Neural spelling correction: translating incorrect sentences to correct sentences for multimedia. *Multimedia Tools and Applications*, pages 1–18.

Chanjun Park, Yeonsu Lee, Chanhee Lee, and Heuiseok Lim. 2020b. Quality, not quantity? : Effect of parallel corpus quantity and quality on neural machine translation. In *The 32st Annual Conference on Human Cognitive Language Technology*, pages 363–368.

Chanjun Park and Heuiseok Lim. 2020. A study on the performance improvement of machine translation using public korean-english parallel corpus. *Journal of Digital Convergence*, 18(6):271–277.

Chanjun Park, Yeongwook Yang, Kinam Park, and Heuiseok Lim. 2020c. Decoding strategies for improving low-resource machine translation. *Electronics*, 9(10):1562.

Matthias Paulik, Sharath Rao, Ian Lane, Stephan Vogel, and Tanja Schultz. 2008. Sentence segmentation and punctuation recovery for spoken language translation. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5105–5108. IEEE.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, CONF. IEEE Signal Processing Society.

Svatava Škodová, Michaela Kuchařová, and Ladislav Šeps. 2012. Discretion of speech units for the text post-processing phase of automatic transcription (in the czech language). In *International Conference on Text, Speech and Dialogue*, pages 446–455. Springer.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*.

Matthew Nicholas Stuttle. 2003. *A Gaussian mixture model spectral representation for speech recognition*. Ph.D. thesis, University of Cambridge.

Jayashri Vajpai and Avnish Bora. 2016. Industrial applications of automatic speech recognition systems. *International Journal of Engineering Research and Applications*, 6(3):88–95.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Kimberly Voll, Stella Atkins, and Bruce Forster. 2008. Improving the utility of speech recognition through error detection. *Journal of digital imaging*, 21(4):371.

Jiangyan Yi, Jianhua Tao, Ye Bai, Zhengkun Tian, and Cunhang Fan. 2020. Adversarial transfer learning for punctuation restoration. *arXiv preprint arXiv:2004.00248*.

Zi-Qiang Zhang, Yan Song, Ming-Hui Wu, Xin Fang, and Li-Rong Dai. 2021. Xlst: Cross-lingual self-training to learn multilingual representation for low resource speech recognition. *arXiv preprint arXiv:2103.08207*.