INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Zero-Shot Joint Modeling of Multiple Spoken-Text-Style Conversion Tasks using Switching Tokens

Mana Ihori, Naoki Makishima, Tomohiro Tanaka,
Akihiko Takashima, Shota Orihashi, Ryo Masumura

NTT Media Intelligence Laboratories, NTT Corporation, Japan
mana.ihori.kx@hco.ntt.co.jp

Abstract

In this paper, we propose a novel spoken-text-style conversion method that can simultaneously execute multiple style conversion modules such as punctuation restoration and disfluency deletion without preparing matched datasets. In practice, transcriptions generated by automatic speech recognition systems are not highly readable because they often include many disfluencies and do not include punctuation marks. To improve their readability, multiple spoken-text-style conversion modules that individually model a single conversion task are cascaded because matched datasets that simultaneously handle multiple conversion tasks are often unavailable. However, the cascading is unstable against the order of tasks because of the chain of conversion errors. Besides, the computation cost of the cascading must be higher than the single conversion. To execute multiple conversion tasks simultaneously without preparing matched datasets, our key idea is to distinguish individual conversion tasks using the *on-off switch*. In our proposed zero-shot joint modeling, we switch the individual tasks using multiple switching tokens, enabling us to utilize a zero-shot learning approach to executing simultaneous conversions. Our experiments on joint modeling of disfluency deletion and punctuation restoration demonstrate the effectiveness of our method.

Index Terms: spoken text style conversion, zero-shot modeling, switching token

1. Introduction

With the rise of various automatic speech recognition (ASR) applications such as smart speakers [1, 2] and automatic dictation systems [3–5], it has become increasingly important to accurately process spoken text that is generated by ASR systems. However, it is difficult to understand the spoken text because it includes many disfluencies and does not involve punctuation marks. Moreover, spoken text adversely affects subsequent natural language processing (e.g., machine translation, summarization, etc.) because these technologies are often developed to handle style-converted text that does not include disfluencies and includes punctuation marks. Therefore, it is important to develop spoken-text-style conversion that converts spoken text into style-converted text. There have been many studies on spoken-text-style conversion, such as disfluency detection [6], capitalization [7], punctuation restoration [8], and inverse text normalization [9]. Since the advent of deep learning, these studies have achieved high performances by preparing a large amount of data that matches each task.

Although the performance of each conversion task has been improved, multiple spoken-text-style conversions should be handled at the same time to improve the readability of spoken text. However, preparing a dataset that handles multiple spoken-style-text conversion tasks simultaneously is costly and time-consuming. Thus, the simplest method of executing multiple tasks simultaneously is to cascade each task. Unfortunately, the cascading is unstable against the order of tasks because of the chain of conversion errors. Besides, the computation cost of the cascading must be higher than the single conversion.

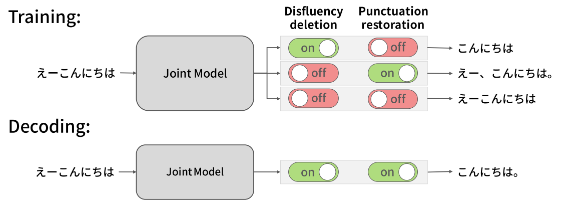

Figure 1: *Zero-shot joint modeling of disfluency deletion and punctuation restoration with the switching.*

To solve these problems, we aim to execute multiple spoken-text-style conversion tasks simultaneously without matched datasets. Our key idea to execute these tasks simultaneously is to introduce the *on-off switch* that can distinguish these individual tasks and a joint conversion task. In the following, we describe how the model learns disfluency deletion and punctuation restoration at the same time. Figure 1 shows an example to execute the disfluency deletion and the punctuation restoration tasks simultaneously. In Figure 1, we prepare the two switches for the disfluency deletion and the punctuation restoration tasks, respectively. When the model learns the disfluency deletion dataset, it recognizes that disfluency deletion is *on* and punctuation restoration is *off*. When the model learns the punctuation restoration, it recognizes that disfluency deletion is *off* and punctuation restoration is *on*. Also, when the model learns not to convert (i.e., the input and output text are the same), it recognizes that disfluency deletion and punctuation restoration are both *off*. In this way, all tasks can be recognized in a single model by switching each task *on* or *off*. Therefore, the model should execute disfluency deletion and punctuation restoration simultaneously if it switches both tasks *on*. We suppose that even if the model does not learn to execute the joint conversion task at all during training, it can execute multiple conversion tasks simultaneously during inference.

In this paper, we propose zero-shot joint modeling of multiple spoken-text-style conversion tasks with a Transformer-based encoder-decoder model by using the switching tokens. The switching tokens switch each conversion task *on* or *off*. In our method, the switching tokens are introduced into the decoder for the input contexts. Our zero-shot joint modeling of multiple conversion tasks starts with two tasks: disfluency deletion and punctuation restoration. For example, in the disfluency deletion task, two switching tokens [`disf_on`] and [`punc_off`] are introduced. The zero-shot joint modeling is learned using the datasets of disfluency deletion and punctuation restoration tasks simultaneously in a single model with the switching tokens. Note that our method does not change the architecture of the conversion model, only adds the switching tokens to the model input. In our experiments using the Corpus of Spontaneous Japanese (CSJ) [10], we compare the results of zero-shot modeling using our method with that of cascading. We demonstrate the effectiveness of the joint modeling of disfluency deletion and punctuation restoration.

2. Related Work

Multiple spoken-text-style conversion tasks: Spoken-text-style conversion has multiple tasks. First, the spoken text has many disfluencies (e.g., fillers and redundant expressions) that are not included in the written text. The disfluency detection task recognizes non-fluent words in the spoken text [6, 11–13]. Moreover, disfluency deletion task removes them from the spoken text in an end-to-end manner [14]. Next, capitalization and punctuation are removed from the spoken text because they do not affect the pronunciation. Therefore, there are tasks that restore capitalization and punctuation for the spoken text [7, 8, 15, 16]. Moreover, ASR systems output numbers, dates, times, and other numerical entities in a literal manner (e.g., 20 $\rightarrow$ twenty). Thus, inverse text normalization was proposed to format entities like numbers, dates, times, and addresses [9, 17, 18]. In addition, a neural generation model of ASR spelling correction [19] and grammar correction [20] was proposed to deal with the many spelling and grammar errors in the spoken text. In this way, the spoken text involves many tasks to be converted, but these tasks have been studied individually.

Spoken-to-written style conversion task: A few studies have been focusing on handling multiple spoken-text-style conversion tasks at the same time. ASR post-processing for readability (APR) was proposed in [21]. APR is the task of converting spoken-style text into written-style text, and this method converts capitalization, disfluency, and grammar errors, as well as properly formatted dates, times, and other numerical entities simultaneously. In this previous study, the dataset to convert these entities simultaneously is prepared using text-to-speech and ASR systems. Moreover, spoken-to-written style conversion was proposed in [22–24]. Spoken-to-written style conversion is the task of converting style unification, postpositional particle expressions restoration, notation correction, punctuation restoration, disfluency deletion, simplification, error correction simultaneously. In this previous study, the dataset is prepared using crowdsourcing. These previous studies need a dataset that can handle multiple tasks simultaneously, but such datasets are not published officially, and it is costly and time-consuming to prepare one.

3. Spoken-Text-Style Conversion with Transformer

This section defines spoken-text-style conversion. In this paper, we model spoken-text-style conversion as a sequence-to-sequence problem in which the source is spoken text and the target is style-converted text. Moreover, we utilize Transformer-based encoder-decoder network architecture [25]. We define the spoken text as $\boldsymbol{X} = \{x_1, \cdots, x_m\}$ and the style-converted text as $\boldsymbol{Y} = \{y_1, \cdots, y_n\}$, where x_m is a token in spoken text and y_n is a token in style-converted text. The Transformer predicts the generation probabilities of a style-converted text $\boldsymbol{Y}$ given a spoken text $\boldsymbol{X}$. The generation probability of $\boldsymbol{Y}$ is defined as

$$P(\boldsymbol{Y}|\boldsymbol{X};\boldsymbol{\Theta}) = \prod_{n=1}^{N} P(y_n|y_{1:n-1}, \boldsymbol{X}; \boldsymbol{\Theta}), \tag{1}$$

where $\boldsymbol{\Theta}$ represents model parameter sets. $P(y_n|y_{1:n-1}, \boldsymbol{X}; \boldsymbol{\Theta})$ can be computed with the Transformer encoder and the Transformer decoder.

Training: We treat the disfluency deletion and punctuation restoration tasks as the spoken-text-style conversion tasks. The disfluency deletion model is trained using a disfluency deletion dataset $\mathcal{D}_{\mathrm{disf}}$. The training loss function $\mathcal{L}_{\mathrm{disf}}$ is defined as

$$\mathcal{L}_{\mathrm{disf}} = -\sum_{(\boldsymbol{X},\boldsymbol{Y})\in\mathcal{D}_{\mathrm{disf}}} \log P(\boldsymbol{Y}|\boldsymbol{X};\boldsymbol{\Theta}_{\mathrm{disf}}). \tag{2}$$

Moreover, the punctuation restoration model is trained using a punctuation restoration dataset $\mathcal{D}_{\mathrm{punc}}$. The training loss function $\mathcal{L}_{\mathrm{punc}}$ is defined as

$$\mathcal{L}_{\mathrm{punc}} = -\sum_{(\boldsymbol{X},\boldsymbol{Y})\in\mathcal{D}_{\mathrm{punc}}} \log P(\boldsymbol{Y}|\boldsymbol{X};\boldsymbol{\Theta}_{\mathrm{punc}}). \tag{3}$$

Cascading decoding: In this paper, we aim to execute a disfluency deletion task and a punctuation restoration task at the same time. When individual conversion tasks are independently modeled from individual datasets, we can execute the joint conversion task via cascading decoding. The decoding problem in the cascading of disfluency deletion and punctuation restoration is defined as

$$\tilde{\boldsymbol{Y}} = \arg\max_{\boldsymbol{Y}} P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\Theta}_{\mathrm{disf}}), \tag{4}$$

$$\hat{\boldsymbol{Y}} = \arg\max_{\boldsymbol{Y}} P(\boldsymbol{Y}|\tilde{\boldsymbol{Y}}, \boldsymbol{\Theta}_{\mathrm{punc}}). \tag{5}$$

Here, the order of cascading can also be reversed. Note that the conversion performance is varied by the order of cascading.

4. Proposed Method

4.1. Strategy

We utilize switching tokens to model multiple spoken-text-style conversion tasks simultaneously without needing to prepare a matched dataset. Figure 2 shows multiple spoken-text-style conversion with Transformer-based encoder-decoder model using the switching tokens. The switching tokens distinguish tasks and represent the *on* state (the target for conversion) or *off* state (not the target for conversion) in each task. For example, if the model is trained with a disfluency deletion dataset, two switching tokens [`disf_on`] and [`punc_off`] are given. Also, if the model is trained with a punctuation restoration dataset, two switching tokens [`disf_off`] and [`punc_on`] are given. Here, "disf" represents disfluency deletion and "punc" represents punctuation restoration.

In this way, even if the model does not train the state of [`disf_on`] [`punc_on`], it should be able to convert two conversion tasks simultaneously during inference. However, the model trained with these two switching patterns ([`disf_on`] [`punc_off`] and [`disf_off`] [`punc_on`]) may be convinced that one task must be *off* if the other task is *on*. Thus, we suppose that the model trained with only these two switching patterns cannot handle the joint conversion task at all. To

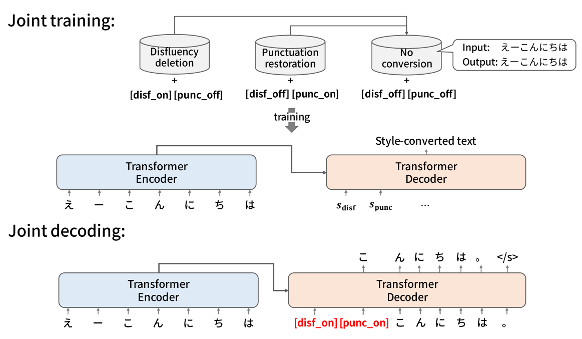

Figure 2: *Proposed method of jointly modeling multiple spoken-text-style conversions.*

make the model learn other states of [`disf_on`] [`punc_off`] and [`disf_off`] [`punc_on`], we make a dataset of the [`disf_off`] [`punc_off`] state. This "no conversion" dataset can be constructed from the disfluency deletion and the punctuation restoration datasets by supposing the input text is not changed by the conversion. We expect that the model understands that one task is not necessarily *off* if the other task is *on* by training with these three datasets. Even if we combine switching tokens that are not trained to the model, we suppose the model can generate the text because it understands the relation of each task and the switching tokens.

4.2. Joint Modeling

The switching tokens are introduced into the decoder for the input contexts, and our model handles disfluency deletion, punctuation restoration, no conversion, and joint conversion tasks as an all-in-one modeling. In other words, our method can learn multiple tasks simultaneously simply by giving the model the switching tokens without changing the model architecture. In spoken-text-style conversion using the switching tokens as context, the generation probability of $\boldsymbol{Y}$ is defined as

$$P(\boldsymbol{Y}|\boldsymbol{X}, s_{\text{disf}}, s_{\text{punc}}; \boldsymbol{\Theta}) = \prod_{n=1}^{N} P(y_n|y_{1:n-1}, \boldsymbol{X}, s_{\text{disf}}, s_{\text{punc}}; \boldsymbol{\Theta}), \quad (6)$$

$$s_{\text{dist}} \in \{[\texttt{disf\_on}], [\texttt{disf\_off}]\}, \quad (7)$$

$$s_{\text{punc}} \in \{[\texttt{punc\_on}], [\texttt{punc\_off}]\}. \quad (8)$$

Here, if we set [`disf_on`] [`punc_off`], [`disf_off`] [`punc_on`], and [`disf_off`] [`punc_off`], the model execute the disfluency deletion task, the punctuation restoration task, and the no conversion task, respectively. Moreover, if we set [`disf_on`] [`punc_on`], the model execute the disfluency deletion task and the punctuation restoration task jointly.

Joint training: In our method, the disfluency deletion dataset $\mathcal{D}_{\text{disf}}$, the punctuation restoration dataset $\mathcal{D}_{\text{punc}}$, and the no conversion dataset $\mathcal{D}_{\text{same}}$ are trained jointly in a single model. The no conversion dataset is made by using the input text of the disfluency deletion dataset and the punctuation dataset. Thus, the model is trained with the sum of twice the amount of disfluency deletion dataset and that of the punctuation restoration dataset. Note that the model is not trained with a matched dataset for the disfluency deletion and punctuation restoration. The training loss function $\mathcal{L}$ is defined as

$$\mathcal{L} = \mathcal{L}_{\text{disf}} + \mathcal{L}_{\text{punc}} + \mathcal{L}_{\text{same}}, \quad (9)$$

$$\mathcal{L}_{\text{disf}} = -\sum_{(\boldsymbol{X},\boldsymbol{Y})\in\mathcal{D}_{\text{disf}}} \log P(\boldsymbol{Y}|\boldsymbol{X}, s_{\text{disf}} = [\texttt{disf\_on}], s_{\text{punc}} = [\texttt{punc\_off}]; \boldsymbol{\Theta}), \quad (10)$$

$$\mathcal{L}_{\text{punc}} = -\sum_{(\boldsymbol{X},\boldsymbol{Y})\in\mathcal{D}_{\text{punc}}} \log P(\boldsymbol{Y}|\boldsymbol{X}, s_{\text{disf}} = [\texttt{disf\_off}], s_{\text{punc}} = [\texttt{punc\_on}]; \boldsymbol{\Theta}), \quad (11)$$

$$\mathcal{L}_{\text{same}} = -\sum_{(\boldsymbol{X},\boldsymbol{Y})\in\mathcal{D}_{\text{same}}} \log P(\boldsymbol{Y}|\boldsymbol{X}, s_{\text{disf}} = [\texttt{disf\_off}], s_{\text{punc}} = [\texttt{punc\_off}]; \boldsymbol{\Theta}). \quad (12)$$

Joint decoding: When the disfluency deletion and punctuation restoration tasks are performed simultaneously, we give the [`disf_on`][`punc_on`] switching tokens to the model. The decoding problem of zero-shot modeling using the switching tokens is defined as

$$\hat{\boldsymbol{Y}} = \arg\max_{\boldsymbol{Y}} P(\boldsymbol{X}|\boldsymbol{Y}, s_{\text{disf}} = [\texttt{disf\_on}], s_{\text{punc}} = [\texttt{punc\_on}]; \boldsymbol{\Theta}). \quad (13)$$

Here, if we feed the switching tokens [`disf_on`] and [`punc_off`] into the model, it performs the disfluency deletion task only. Moreover, if we feed the switching tokens [`disf_off`] and [`punc_on`] into the model, it performs the punctuation restoration task only.

5. Experiments

5.1. Datasets

We evaluated effectiveness of our method using the CSJ dataset [10]. We divided the CSJ into a training set, a validation set, and a test set. We used disfluency deletion and punctuation restoration for the spoken-text-style conversion task. Thus, we divided a training set and a validation set for each task. First, in the punctuation restoration task, we made pair data of the spoken text and the text with punctuation marks restored using crowdsourcing. We prepared a 50,000-sentence training set and a 5,000-sentence validation set for the punctuation restoration task. Next, in the disfluency deletion task, we made pair data of the spoken text and the text with fillers removed based on the part of speech in the CSJ. Here, we prepared a 50,000-sentence training set and a 5,000-sentence validation set to prevent imbalanced datasets from affecting the conversion performance. Finally, we prepared a test set for each task and it consists of 3,949 sentences. In addition, we investigated how performance differed depending on the amount of data in the training set by dividing the training set into 10,000, 30,000, and 50,000 sentences, respectively.

5.2. Setups

For evaluation purposes, we constructed three Transformer based encoder-decoder models: 1) a disfluency deletion model, 2) a punctuation restoration model, and 3) a switching token-based joint model. The disfluency deletion and punctuation restoration models were trained using only each dataset without the switching tokens. The switching token-based joint model was our proposed model trained with both datasets. Note that the Transformer architecture was the same in these models.

Table 1: *Results of disfluency deletion and punctuation restoration, respectively.*

	Task		10k sentences in each dataset			30k sentences in each dataset			50k sentences in each dataset		
	disf	punc	BLEU	METEOR	GLEU	BLEU	METEOR	GLEU	BLEU	METEOR	GLEU
1).	✓	-	0.879	0.981	0.831	0.901	0.988	0.858	0.909	0.991	0.870
3).	on	off	0.892	0.985	0.849	0.902	0.987	0.864	0.905	0.990	0.864
2).	-	✓	0.767	0.945	0.644	0.809	0.962	0.693	0.816	0.964	0.700
3).	off	on	0.838	0.981	0.725	0.839	0.982	0.727	0.844	0.983	0.735

1). disfluency deletion model 2). punctuation restoration model 3). switching token-based joint model

Table 2: *Results of multiple spoken-text-style conversion tasks.*

	Task		Speed	Memory	10k sentences in each dataset			30k sentences in each dataset			50k sentences in each dataset		
	disf	punc			BLEU	METEOR	GLEU	BLEU	METEOR	GLEU	BLEU	METEOR	GLEU
1+2).	cascading (→)		× 2.0	× 2.0	0.765	0.944	0.668	0.824	0.969	0.730	0.839	0.974	0.757
	cascading (←)		× 2.0	× 2.0	0.683	0.889	0.545	0.750	0.923	0.622	0.764	0.929	0.640
3).	cascading (→)		× 2.0	× 1.0	0.805	0.972	0.712	0.815	0.974	0.730	0.820	0.977	0.738
	cascading (←)		× 2.0	× 1.0	0.722	0.898	0.596	0.738	0.904	0.609	0.733	0.905	0.608
	on	on	× 1.0	× 1.0	0.799	0.972	0.704	0.814	0.974	0.726	0.818	0.973	0.725

1). disfluency deletion model 2). punctuation restoration model 3). switching token-based joint model

We employed the following configurations. In the encoder, a 4-layer Transformer encoder block with 512 units was introduced. In the decoder, a 2-layer Transformer decoder block with 512 units was introduced. The output unit size (corresponding to the number of tokens in the training and validation sets) was set to 3,316. To train the Transformer, we used the RAdam optimizer [26] and label smoothing [27] with a smoothing parameter of 0.1. We set the mini-batch size to 64 sentences and the dropout rate in each Transformer block to 0.1. All trainable parameters were randomly initialized, and we used characters as tokens. For the decoding, we used a beam search algorithm in which the beam size was set to 4. For the evaluation, we calculated the automatic evaluation scores using three metrics: BLEU [28], METEOR [29], and GLEU [30]. BLEU and METEOR are metrics used in machine translation tasks, and they compute the matching of the reference and hypothesis text. GLEU is a metric used in grammatical error correction tasks [31], and it is computed using the input, reference, and hypothesis text. If the model generates n-gram that is not in the input text but is in the reference text, the evaluation score becomes high. In this paper, we use 4-gram for BLEU and GLEU.

5.3. Results

Table 1 shows the respective evaluation results of the disfluency deletion task and the punctuation restoration task. Moreover, Table 2 shows the evaluation results of cascading decoding with the disfluency deletion and punctuation restoration models, cascading decoding with switching token-based joint model, and the joint decoding with the joint model. Cascading (→) in Table 2 represents the order of cascading, and cascading (→) was cascaded in the order of the disfluency deletion task to the punctuation restoration task. Cascading (←) was processed in the reverse order. In Table 2, the joint modeling where each switch is *on* is our proposed zero-shot joint decoding.

First, we focus on the results in Table 1. In the disfluency deletion task, when the amount of training data was 50,000 sentences, the disfluency deletion model slightly outperformed our method. Otherwise, our method outperformed the disfluency deletion model. Moreover, in the punctuation restoration task, our method substantially outperformed the the punctuation restoration model. This indicates that joint modeling using datasets of all tasks improves the performance of each task. Next, we focus on the results in Table 2. The cascading results show that the performance was substantially different based on the cascading order in all models. The result of cascading (→) substantially outperformed the result of cascading (←). The reason is that the model replaced punctuation marks that were generated in the punctuation restoration task with other words because it does not learn to generate punctuation marks in the disfluency deletion task. Here, the result of cascading (←) using our method substantially underperformed the result using baseline because our method has high performance of punctuation restoration task and many punctuation marks were replaced with other words by cascading. Therefore, in cascading, it is difficult to convert spoken text robust because the order of processing has a large effect on performance.

In addition, we focus on the results of the zero-shot joint decoding in Table 2. When the training data was 10,000 sentences, the zero-shot joint decoding outperformed the cascading decoding with the disfluency deletion and punctuation restoration models. Moreover, when the training data was 10,000 and 30,000 sentences, the zero-shot joint decoding performance was comparable with cascading decoding with switching token-based joint model. On the other hand, when there were 50,000 sentences in each dataset, the cascading (→) with the disfluency deletion and punctuation restoration models slightly outperformed the zero-shot joint decoding because influence against chain of conversion errors was mitigated. This indicates that if the amount of data in each dataset is small, the performance of the zero-shot joint decoding is effective to prevent the chain of conversion errors. These results confirm that our proposed zero-shot joint decoding with switching token joint modeling are stable approach and can achieve comparable performance to cascading decoding with the best choice of the processing order although matched datasets for the joint conversion problem are unavailable. Moreover, the proposed joint decoding is twice as fast as the cascading decoding while keeping the superior performance.

6. Conclusions

In this paper, we proposed a spoken text conversion method that can simultaneously execute multiple style conversion modules without needing to prepare a matched dataset. Our key idea is to distinguish individual conversion tasks using the *on-off switch* with multiple switching tokens. The switching tokens enable us to utilize a zero-shot learning approach to executing simultaneous conversions. In the evaluation experiment, we treated disfluency deletion and punctuation restoration tasks as multiple spoken-text-style conversion tasks. The results demonstrated that our method can execute multiple conversion tasks simultaneously that is twice as fast as the cascading while keeping superior performance even though matched training dataset was not used at all.

7. References

[1] B. Li, T. Sainath, A. Narayanan, J. Caroselli, M. Bacchiani, A. Misra, I. Shafran, H. Sak, G. Pundak, K. Chin, K. C. Sim, R. J. Weiss, K. Wilson, E. Variani, C. Kim, O. Siohan, M. Weintraub, E. McDermott, R. Rose, and M. Shannon, "Acoustic modeling for google home," in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2017.

[2] A. Purington, J. Taft, S. Sannon, N. N. Bazarova, and S. Taylor, ""alexa is my new bff": Social roles, user satisfaction, and personification of the amazon echo," in *Proc. Conference Extended Abstracts on Human Factors in Computing Systems (CHI)*, 2017, pp. 2853–2859.

[3] G. Shang, W. Ding, Z. Zhang, A. J.-P. Tixier, P. Meladianos, M. Vazirgiannis, and J.-P. Lorré, "Unsupervised abstractive meeting summarization with multi-sentence compression and budgeted submodular maximization," in *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

[4] M. Li, L. Zhang, H. Ji, and R. J. Radke, "Keep meeting summaries on topic: Abstractive multi-modal meeting summarization," in *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019, pp. 2190–2196.

[5] Z. Zhao, H. Pan, C. Fan, Y. Liu, L. Li, M. Yang, and D. Cai, "Abstractive meeting summarization via hierarchical adaptive segmental network learning," in *Proc. the World Wide Web Conference (WWW)*, 2019, pp. 3455–3461.

[6] Q. Dong, F. Wang, Z. Yang, W. Chen, S. Xu, and B. Xu, "Adapting translation models for transcript disfluency detection," in *Proc. of the Conference on Artificial Intelligence (AAAI)*, 2019, pp. 6351–6358.

[7] B. Nguyen, V. B. H. Nguyen, H. Nguyen, P. N. Phuong, T.-L. Nguyen, Q. T. Do, and L. C. Mai, "Fast and accurate capitalization and punctuation for automatic speech recognition using transformer and chunk merging," in *Proc. Conference of the Oriental International Committee for the Co-ordination and Standardisation of Speech Databases and Assessment Techniques (O-COCOSDA)*, 2019, pp. 1–5.

[8] S. Kim, "Deep recurrent neural networks with layer-wise multi-head attentions for punctuation restoration," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 7280–7284.

[9] E. Pusateri, B. R. Ambati, E. Brooks, O. Platek, D. McAllaster, and V. Nagesha, "A mostly data-driven approach to inverse text normalization." in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2017, pp. 2784–2788.

[10] K. Maekawa, H. Koiso, S. Furui, and H. Isahara, "Spontaneous speech corpus of japanese," in *Proc. International Conference on Language Resources and Evaluation (LREC)*, 2000, pp. 947–9520.

[11] N. Bach and F. Huang, "Noisy BiLSTM-Based Models for Disfluency Detection," in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2019, pp. 4230–4234.

[12] T. Tanaka, R. Masumura, T. Moriya, T. Oba, and Y. Aono, "Disfluency detection based on speech-aware token-by-token sequence labeling with blstm-crfs and attention mechanisms," in *Proc. Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2019, pp. 1009–1013.

[13] S. Wang, W. Che, Y. Zhang, M. Zhang, and T. Liu, "Transition-based disfluency detection using lstms," in *Proc. the Conference on Empirical Methods in Natural Language Processing (EMLP)*, 2017, pp. 2785–2794.

[14] M. Ihori, A. Takashima, and R. Masumura, "Large-context pointer-generator networks for spoken-to-written style conversion," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 8189–8193.

[15] O. Tilk and T. Alumäe, "Bidirectional recurrent neural network with attention mechanism for punctuation restoration." in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2016, pp. 3047–3051.

[16] M. Á. Tündik, B. Tarjan, and G. Szaszák, "A bilingual comparison of maxent-and rnn-based punctuation restoration in speech transcripts," in *Proc. the International Conference on Cognitive Infocommunications (CogInfoCom)*, 2017, pp. 121–126.

[17] Y.-C. Ju and J. Odell, "A language-modeling approach to inverse text normalization and data cleanup for multimodal voice search applications," in *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2008, pp. 2179–2182.

[18] M. Sunkara, C. Shivade, S. Bodapati, and K. Kirchhoff, "Neural inverse text normalization," *arXiv preprint arXiv:2102.06380*, 2021.

[19] J. Guo, T. N. Sainath, and R. J. Weiss, "A spelling correction model for end-to-end speech recognition," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 5651–5655.

[20] O. Hrinchuk, M. Popova, and B. Ginsburg, "Correction of automatic speech recognition with transformer sequence-to-sequence model," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7074–7078.

[21] J. Liao, S. E. Eskimez, L. Lu, Y. Shi, M. Gong, L. Shou, H. Qu, and M. Zeng, "Improving readability for automatic speech recognition transcription," *arXiv preprint arXiv:2004.04438*, 2020.

[22] M. Ihori, A. Takashima, and R. Masumura, "Parallel corpus for Japanese spoken-to-written style conversion," in *Proc. Language Resources and Evaluation Conference (LREC)*, 2020, pp. 6346–6353.

[23] M. Ihori, R. Masumura, N. Makishima, T. Tanaka, A. Takashima, and S. Orihashi, "Memory attentive fusion: External language model integration for transformer-based sequence-to-sequence model," in *Proc. the International Conference on Natural Language Generation (INLG)*, 2020, pp. 1–6.

[24] M. Ihori, N. Makishima, T. Tanaka, A. Takashima, S. Orihashi, and R. Masumura, "Mapgn: Masked pointer-generator network for sequence-to-sequence pre-training," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 7563–7567.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Advances in neural information processing systems (NIPS)*, 2017, pp. 5998–6008.

[26] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," in *Proc. International Conference on Learning Representations (ICLR)*, 2019.

[27] M. Lukasik, S. Bhojanapalli, A. Menon, and S. Kumar, "Does label smoothing mitigate label noise?" in *Proc. the International Conference on Machine Learning (ICML)*, 2020, pp. 6448–6458.

[28] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proc. Annual Meeting on Association for Computational Linguistics (ACL)*, 2002, pp. 311–318.

[29] S. Banerjee and A. Lavie, "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments," in *Proc. the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 2005, pp. 65–72.

[30] C. Napoles, K. Sakaguchi, M. Post, and J. Tetreault, "Ground truth for grammatical error correction metrics," in *Proc. Annual Meeting on Association for Computational Linguistics (ACL)*, 2015, pp. 588–593.

[31] C. Napoles, M. Nădejde, and J. Tetreault, "Enabling robust grammatical error correction in new domains: Data sets, metrics, and analyses," *Transactions of the Association for Computational Linguistics*, pp. 551–566, 2019.