Disfluency Disentanglement Enhancement in Spoken-Text-Style Transfer for Spontaneous Speech Synthesis

Yuuto Nakata*, Daiki Yoshioka†, Wen-Chin Huang†, Tomoki Toda†
* National Institute of Technology, Tokuyama College, Japan
† Nagoya University, Japan

***Abstract*—Spoken-text-style transfer (STST) aims to convert a given text to a desired style while preserving its semantic content. In this paper, we focus on modeling the disfluency in spoken text, which can be useful as a preprocessing step in spontaneous speech synthesis. Previous methods suffer from unknown words and confusion between normal and disfluent words. Our proposed solutions to the above-mentioned problems include a masked language model (MLM) approach to temporally replace unknown words, and a disfluency symbol representation (DSR) to increase the discrepancy against normal words. Experimental evaluation results show that MLM improves the robustness against unknown words, and the use of DSR achieves a higher transfer accuracy. Finally, we show the potential of speaker-modeling to achieve speaker-wise disfluency control in STST.**

I. Introduction

Text-to-speech (TTS) systems based on deep neural networks have been shown to be capable of synthesizing speech with human-like naturalness [1]. Such systems, however, are mostly trained with datasets containing read speech, particularly collected from audiobooks [2], [3]. Such datasets are often recorded in a controlled setting, with a stable speaking rate, a neutral emotion, and a fixed prompt. In contrast, spontaneous speech synthesis poses a completely different challenge [4]. While there yet exists a formal definition of spontaneous speech, the correct modeling of multiple different dimensions is required. Efforts have been made to investigate these factors, including prosody variations [5], [6], breathing [7] and vocal efforts [8].

Disfluency modeling has been heavily studied among the important factors in spontaneous speech synthesis, as researchers have shown its importance in spontaneous speech, on both the speaker side [9], [10] and the listener side [11], [12]. Disfluency refers to a collection of speech phenomena, including repairs, repetitions, lengthening, and discourse marker [13]. One crucial phenomenon is called the filler, which is a non-lexical sound used to express a pause in thinking [10]. While there have been many successful attempts to augment modern TTS models with the ability to synthesize fillers [13], [14], the integration of such models with, for instance, conversational agents, still remains a challenge, as stated in [15]. The reason is that most dialogue systems are trained with written texts, thus the generation process of fillers given a written text is needed.

TABLE I
Examples of disfluent words used in this work. あの, えー, まー are roughly equivalent to "umm", "uhh", "well", respectively. The slip word ふっ has the pronunciation "fu", which stems from the pronunciation of the next word 普段, "fudan". The slip word ん has the pronunciation "n".

Text example	Filler	Slip
本当に あのー ふっ 普段 は 気が付か ない で	あのー	ふっ
えー 調査 を 実施 し て い ない 自治体 が	えー	
よく まー あの ん 声 が	まー, あの	ん

Spoken-text-style transfer (STST) has been studied as a solution to the above-mentioned problem. STST can be defined as a task that converts a given (usually written) text into its spoken version while preserving its meaning. However, training an STST model requires a large-scale parallel text corpus, which is difficult to collect. To address this issue, previous work proposed a framework based on conditional variational autoencoders (CVAEs) [16] which enabled STST model training with non-parallel data, which is more accessible [17].

Despite its effectiveness in STST, a remaining problem of the CVAE method is the disentanglement of disfluency. In this work, we are particularly interested in the modeling of two kinds of *disfluent words*: fillers and slips. The definition of filler has been given above, and slips refer to a temporary speaking mistake when pronouncing a word. Table I shows some examples. On the other hand, content words refer to words that carry semantic meaning and function independently [18]. Thus, disfluency disentanglement refers to the task of distinguishing between disfluent words. This can be especially difficult when the input contains unknown words. Furthermore, as shown in [19], the modeling of disfluency can be highly speaker-dependent, highlighting the importance of joint disfluency and speaker modeling in STST.

This paper presents an STST system with enhanced disfluency disentanglement ability. First, we propose a masked language modeling (MLM)–based approach to temporarily replace unknown words during the style transfer process. We then propose several disfluency symbol representations (DSRs) and identify the optimal representation to improve style control performance. Finally, we investigate the potential of speaker modeling in STST to capture the influence of speaker

979-8-3315-7206-8/25/$31.00

identity when modeling disfluency styles. Experfiments were conducted on the corpus of spontaneous Japanese (CSJ) [20], a spontaneous Japanese corpus, and experimental evaluation results demonstrated the effectiveness of the proposed techniques.

II. The CVAE baseline

In this section, we introduce an STST baseline model based on CVAE, which enables training using non-parallel data [17]. Assume we have a non-parallel dataset containing samples with their style label. In this work, each sample point is a tuple $(\boldsymbol{x}, \boldsymbol{s})$ where $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$ is an input word sequence of length N, and $\boldsymbol{s} \in \{(1,0),(0,1)\}$ is a two-dimensional one-hot vector representing fluent or disfluent. During training, given the input $\boldsymbol{x}$, the encoder first encodes a content feature vector $\boldsymbol{z}$. Then, together with the style label $\boldsymbol{s}$, the decoder generates a target word sequence $\hat{\boldsymbol{x}} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_M)$ of length M. The training objective is to maximize the following variational lower bound L:

$$L = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\hat{\boldsymbol{x}}|\boldsymbol{z}, \boldsymbol{s})] - \mathrm{KL}\left[q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\right], \quad (1)$$

where $q(\boldsymbol{z}|\boldsymbol{x})$ represents the approximate posterior distribution encoding $\boldsymbol{x}$ into $\boldsymbol{z}$ via the encoder, $p(\hat{\boldsymbol{x}}|\boldsymbol{z}, \boldsymbol{s})$ is the output probability distribution of $\hat{\boldsymbol{x}}$ reconstructed from $\boldsymbol{z}$ and $\boldsymbol{s}$ via the decoder, and $p(\boldsymbol{z})$ denotes the prior distribution of $\boldsymbol{z}$. During inference, given an input text sequence, style transfer can be done by providing a style label with the desired direction (fluent to disfluent or disfluent to fluent).

The encoder is based on a bidirectional Long Short-Term Memory (LSTM) network [21], and the decoder is an auto-regressive unidirectional LSTM, where at each time step t, the decoder generates the hidden state given the hidden state of the last time step $\boldsymbol{h}_{t-1}$ and the generated word at the last time step $\hat{x}_{t-1}$:

$$\boldsymbol{h}_t = \mathrm{LSTM}(\boldsymbol{h}_{t-1}, \hat{x}_{t-1}). \quad (2)$$

Content word storage. To ensure that the content words are preserved during the encoding-decoding process, an attention-based content word storage mechanism was proposed. Specifically, given the input word sequence, the MeCab[1] Japanese text analyzer toolkit was used to infer the part-of-speech of each input word, and the noun, verb, adverb, and adjective words are concatenated to form the content word sequence $\boldsymbol{x}_{\mathrm{CW}}$. Then, a positional encoding is added to each word [22] to form the content word representation sequence $\boldsymbol{e}_{\mathrm{CW}}$. During the decoding process, at each time step t, an attention weight $\boldsymbol{w}_t$ is first computed, from which the context vector $\boldsymbol{c}_t$ is derived:

$$\boldsymbol{w}_t = \mathrm{Softmax}(\boldsymbol{e}_{\mathrm{CW}} \odot \boldsymbol{h}_t), \quad (3)$$

$$\boldsymbol{c}_t = \boldsymbol{e}_{\mathrm{CW}} \odot \boldsymbol{w}_t. \quad (4)$$

Finally, using $\boldsymbol{h}_t$ and $\boldsymbol{c}_t$, the decoder outputs a probability distribution $\boldsymbol{p}_t$, which can be used to sample the output word sequence:

$$\boldsymbol{p}_t = \mathrm{Softmax}(\mathrm{Linear}([\boldsymbol{h}_t, \boldsymbol{c}_t])). \quad (5)$$

[1] https://github.com/SamuraiT/mecab-python3

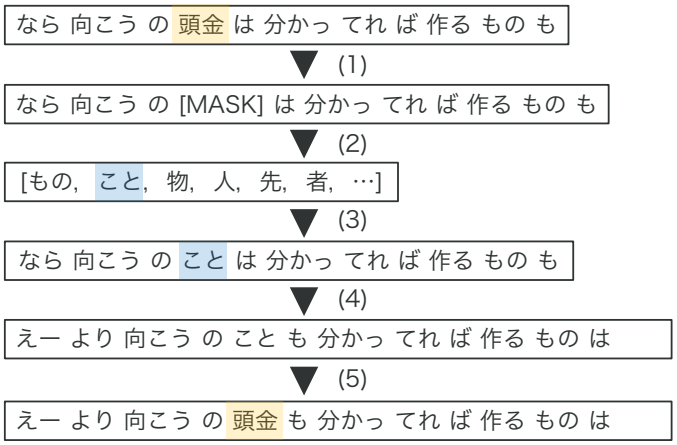

Fig. 1. Text style transfer flow with the proposed masked language modeling technique for unknown word handling. The word in yellow, 頭金, means "down payment" in English. The word in blue, こと, means "thing" in English.

III. Proposed Method

Our proposed method is a collection of techniques to improve the disfluency disentanglement ability of the CVAE baseline, which will be described in the following subsections.

A. Handling unknown words with masked language modeling

One of the major problems in the CVAE baseline is the inability to handle unknown words. Specifically, when encountered with input text containing words not present in the training data, the model tends to delete content words and generate unnecessary fillers and repetitions. To address this problem, we propose an MLM-based approach. As shown in Figure 1, for each unknown word in the input text sequence, our proposed method performs the following five steps:

1) Replace the unknown word with a pre-defined [MASK] token.
2) Given the output of step (1), generate a list of possible output words at the [MASK] position using an MLM.
3) Sample a word from the candidate list such that (a) the sampled word is a known word, and (b) the sampled word is not in the input text sequence. Then, replace the [MASK] token with the sampled word.
4) Perform STST using the output text sequence from step (3).
5) Identify the sampled word and restore the original unknown word.

The idea is to temporarily replace unknown words with known words, i.e., words that appeared in the training data of the STST model, so that the model can correctly recognize the input text and perform STST. The replacement process involves using MLMs like BERT [23], whose masked language modeling training objective was naturally suitable for this task.

B. Disfluency symbol representations (DSRs)

To improve the disfluency disentanglement ability of the model, we propose and design several DSRs to explicitly

TABLE II
EXAMPLES OF THE DISFLUENCY SYMBOL REPRESENTATIONS (DSRS) INVESTIGATED IN THIS WORK. ほー IS ROUGHLY EQUIVALENT TO "I SEE". THE SLIP WORD で HAS THE PRONUNCIATION "DE", WHICH STEMS FROM THE PRONUNCIATION OF THE NEXT WORD 電話, "DENWA".

DSR Type	Text Example
Raw	ほー そういう で 電話 の し方 も
Annotated	＜ほー＞ そういう （で） 電話 の し方 も
s-Annotated	＜ ほー ＞ そういう （ で ） 電話 の し方 も
Symbolic	[FILLER] そういう [SLIP] 電話 の し方 も

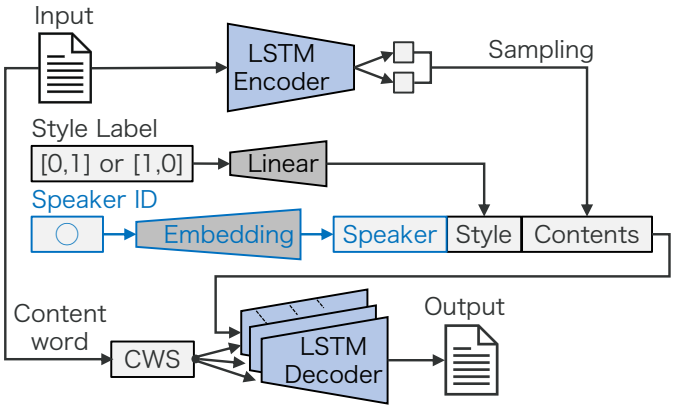

Fig. 2. Model structure with applied speaker embedding.

distinguish between content words and disfluent words. Table II shows the DSRs that are investigated in this study, which we explain below.

- ***Raw***. This refers to the original text.
- ***Annotated***. Here, fillers and slips are enclosed in symbols like brackets and parentheses and are treated as single words. (For instance, ＜ほー＞, （で）). After style transfer, the brackets and parentheses are removed.
- ***s-Annotated***. This is similar to ***Annotated***, with spaces inserted between the word and the brackets or parentheses. (for instance, ＜ ほー ＞, （ で ）). Again, after style transfer, the brackets and parentheses are removed. The motivation is to give the model more flexibility.
- ***Symbolic***. Here, fillers and hesitations are replaced with predefined symbols, such as [FILLER] and [SLIP]. After style transfer, these symbols are replaced with the most common filler and slip words, which are "えー" (English: "uhh") and "ん" (English pronunciation: "n"), respectively.

In the experiments, we verify the effectiveness of each DSR and identify the optimal choice for the task of STST.

C. Speaker modeling

Finally, we investigate the effectiveness of speaker modeling in the context of STST by introducing speaker embeddings in the CVAE model. Figure 2 illustrates the CVAE model augmented with speaker embeddings. Specifically, we represent the speaker information with a one-hot speaker ID vector, which is then sent into an embedding lookup table to obtain a speaker embedding vector. This speaker embedding is then concatenated with the content feature vector and the style label, and subsequently sent into the decoder.

IV. EXPERIMENTAL SETTING

A. Data

We used the CSJ corpus [20] in our experiments. CSJ contains live recordings of academic presentations in nine different research fields and studio recordings of non-professionals speaking on everyday topics in front of a small audience in a relaxed atmosphere. Each recording in the CSJ corpus has human transcription, with disfluent word annotations. The following is an example: 僕 は (filler:えー) ラーメン が (slip:す) 好き だ[2]. However, the recordings are structured at the lecture level, so the following preprocessing steps are applied to obtain sentence-level text:

- Segment each lecture transcript into utterance units based on voice activity detection.
- Divide the utterance units into shorter segments of approximately ten to thirty words.
- Annotate each short segment as fluent or disfluent based on the presence or absence of disfluent words.

In total, there are 318,055 disfluent samples and 336,762 fluent samples in the training set, 8,559 disfluent samples and 8,663 fluent samples in the validation set, and 4,036 disfluent samples and 4,558 fluent samples in the testing set.

We assigned an ID to each lecture and used them as the speaker ID in the speaker modeling experiments, resulting in 3244 speaker IDs. Through our manual inspection, there exists a small amount of repeated speaker IDs, but should not affect the validity of the experiment. For the MLM, we used a publicly available pre-trained Japanese BERT [23] model[3].

B. Evaluation Metrics

- **Accuracy (AC)** refers to the accuracy of a convolutional neural network-based style classifier [24] trained on the dataset to distinguish between disfluent and fluent text. This metric evaluates the style transfer performance.
- **BLEU** [25] score is a commonly used metric in machine translation to evaluate content preservation. In our experiments, we compare the converted text with the reference text to evaluate the content preservation ability of the model.
- **Content word error rate (CWER)** is an extended version of the word error rate metric which is commonly used in the speech recognition task. This metric also evaluates the content preservation ability of the model by calculating the WER between the content words in the input text and the converted text.
- **Disfluent word error rate (DWER)** is similar to the CWER described above, with the difference in that

[2]えー is roughly equivalent to "uhh". The slip word す has the pronunciation "su", which stems from the pronunciation of the next word 好き, "suki".
[3]https://www.nlp.ecei.tohoku.ac.jp/news-release/3284/

TABLE III
EXPERIMENTAL EVALUATION RESULTS. FOR DSR TYPE "RAW", SINCE WE CANNOT EXTRACT DISFLUENT WORDS AFTER TRANSFER, THE CALCULATION DWER IS NOT APPLICABLE, THUS SHOWN AS "N/A".

Index	MLM	DSR	Speaker modeling	DSR Type	AC [%] ↑	BLEU ↑	CWER [%] ↓	DWER [%] ↓
(1)				Raw	61.31	57.65	8.73	N/A
(2)	✓			Raw	56.30	59.09	7.85	N/A
(3)	✓	✓		Annotated	79.19	58.73	9.21	94.25
(4)	✓	✓		s-Annotated	77.58	50.32	14.11	91.49
(5)	✓	✓		Symbolic	95.07	58.50	7.19	87.80
(6)	✓	✓	✓	Annotated	78.38	58.95	9.33	90.17
(7)	✓	✓	✓	Symbolic	95.28	58.10	6.92	87.07

DWER calculates the WER between the disfluent word sequence in the converted text and the reference text. Compared to the AC metric, which does not consider speaker-wise characteristics, the calculation of DWER directly operates on the reference text, thus is regarded as the main metric to evaluate the model's ability to perform speaker-dependent disfluency transfer.

V. EXPERIMENTAL EVALUATION RESULTS

In the experiments, we conducted STST in two directions and showed the averaged results: (1) fluent → disfluent, and (2) disfluent → fluent. Note that for DWER, we only showed the fluent → disfluent results.

A. Effectiveness of MLM

Table III presents the experimental evaluation results. We first examine the effectiveness of the use of MLM. By comparing row (2) to row (1), we observe degradation in AC, the style transfer performance metric. However, we observed improvements in BLEU and CWER, the content preservation ability metrics (with $p < 0.01$ in a paired t-test). This suggests that the introduction of the MLM improves the model's robustness to unknown words.

B. Effectiveness of DSR

Next, we examine the effectiveness of DSR. From Table III, by comparing rows (3), (4), and (5) to row (2), we can observe a substantial improvement in AC (all with $p < 0.01$ in a paired t-test). This result indicates that explicitly using DSRs to distinguish between content words and disfluent words improves the disfluency disentanglement of the model. Among the three DSR types, s-Annotated suffered from severe degradation in BLEU and CWER. This is likely due to a reduced content preservation ability caused by space insertion. Therefore, Annotated and Symbolic are considered suitable DSR formats for improving style transfer performance while maintaining the content preservation ability.

C. Effectiveness of speaker modeling

1) Quantitative study: From Table III, by comparing rows (6) and (7) to rows (3) and (5), respectively, we can observe improvements in the DWER metric (both with $p < 0.01$ in a paired t-test). Table IV further shows the deletion, insertion, and substitution errors in the DWER calculation. We can observe that introducing speaker modeling can reduce substitution errors (both with $p < 0.01$ in a paired t-test). On the other hand, the insertion and deletion errors remained relatively unchanged: a paired t-test showed a significant difference for the annotated DSR type ($p < 0.01$), while no statistically significant change was observed for the symbolic DSR type. This suggests that the model augmented with speaker modeling ability is capable of performing speaker-dependent disfluency transfer.

TABLE IV
BREAKDOWN OF DELETION PLUS INSERTION (D+I) AND SUBSTITUTION (S) ERRORS IN THE DISFLUENT WORD ERROR RATE CALCULATION.

Speaker modeling	DSR Type	D+I ↓	S ↓
	Annotated	1786	1282
✓	Annotated	1725	1210
	Symbolic	1361	1497
✓	Symbolic	1372	1462

TABLE V
STYLE TRANSFER EXAMPLES BY CONDITIONING THE MODEL WITH TWO RANDOMLY CHOSEN SPEAKER IDS (SIDS) FROM THE TRAINING SET. THE FILLER WORDS ARE UNDERLINED. あの, えー, ARE ROUGHLY EQUIVALENT TO "UMM", "UHH", RESPECTIVELY.

Input	こちらのスライドでは信号処理を用いて 音声をどのように加工するかについて説明します
Output (SID: A)	あのーこちらのスライドでは信号処理を用いて 音声をようにあのー加工するかについて説明します
Output (SID: B)	えーこちらのスライドでは信号処理を用いて 音声をどのように加工するについて説明しますがえー

2) Qualitative study: Table V shows style transfer examples when conditioning with different speaker IDs. The most common filler in the training data of speakers A and B were

When comparing the effectiveness of using different DSR types in speaker modeling, we observed that when using the annotated DSR type, the DWER improved from 94.25% to 90.17%, while the symbolic DSR type yielded a marginal improvement, from 87.80% to 87.07%. A possible explanation is that when using the symbolic DSR type, all disfluent words are replaced with the predefined symbols dirst, then restored with the most common disfluent words. This procedure may reduce its effectiveness for speaker modeling.

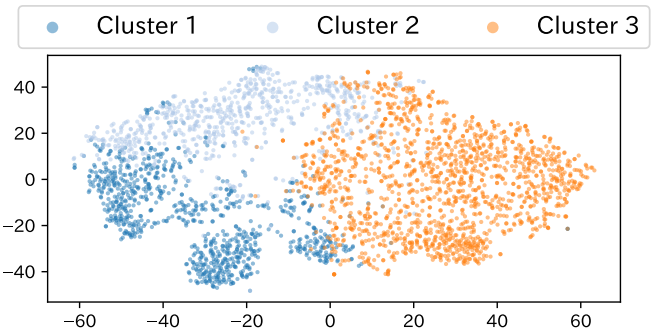

Fig. 3. The t-SNE visualization of the 3244 speaker embeddings learned in the proposed model. Each point represents a speaker embedding, and colors indicate cluster assignments obtained via k-means clustering with three clusters.

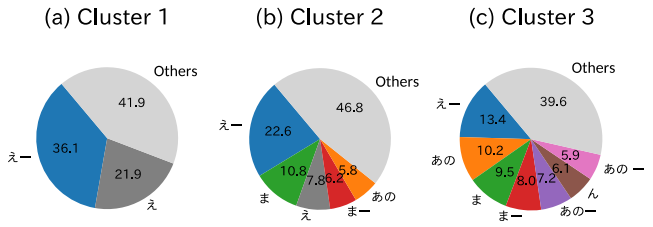

Fig. 4. Disfluent word distribution of each speaker cluster in Figure 3. All disfluent words with a percentage less than 5% are categorized as "others".

"あのー" (English: "umm") and "えー" (English: "uhh"), respectively. As shown in the table, the model successfully inserted proper fillers for each speaker, demonstrating its ability to model disfluency in a speaker-dependent manner.

We further analyzed the speaker space in the model of row (6) in Table III. Recall that the training set included 3,244 speaker IDs, each associated with an embedding learned during training. We applied k-means clustering [26] with three clusters, and then projected these embeddings into a two-dimensional space using t-SNE [27]. Each speaker was assigned a cluster ID based on the learned k-means model. For each cluster, we collected all disfluent words spoken by its speakers and computed the distribution of disfluencies within the cluster.

The results are shown in Figures 3 and 4. We can see that in cluster 1, most fillers start with "え" (English: "uh"), and in cluster 2, the variation of disfluent words increases, and cluster 3 has the largest variation. This shows that the speaker embedding space captures the disfluent word distribution.

VI. Conclusion and Discussions

In this work, we proposed several techniques to enhance the disfluency disentanglement ability of the previously proposed STST method. The techniques include an MLM for robust style transfer against unknown words, DSRs for improving discrepancy between content words and disfluent words, as well as speaker modeling for enabling speaker-dependent disfluency transfer. Experiments were conducted on the CSJ corpus, and the results confirmed the effectiveness of the proposed methods.

Readers might wonder that, in the context of the task of interest, the CVAE-based method can be easily outperformed using a large language model (LLM). We have, in fact, conducted such a comparison in a separate paper [17]. To briefly summarize the results, the CVAE method outperformed GPT 3.5 in terms of style transfer and content preservation ability, and was on par or inferior to GPT 4. However, most importantly, the CVAE method was much faster than both GPT 3.5 and GPT 4. Considering that the ultimate goal is to combine the model with a downstream TTS engine, we concluded that the CVAE-based method has its advantages over LLM-based methods.

Another potential concern is the lack of integrated evaluation with a downstream TTS system. While we agree this is an important direction, we have, in fact, already conducted such experiments, and the results are summarized in a separate manuscript, which is currently under peer review. Due to double submission policies, we are unable to include those findings here, but we plan to make them publicly available upon acceptance.

Acknowledgment

This work was supported in part by JST CREST Grant Number JPMJCR22D1, Japan, and JSPS KAKENHI Grant Number 21H05054.

References

[1] J. Shen, R. Pang, R. J. Weiss, *et al.*, "Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions," in *Proc. ICASSP*, 2018, pp. 4779–4783.

[2] K. Ito and L. Johnson, *The LJ Speech Dataset*, https://keithito.com/LJ-Speech-Dataset/, 2017.

[3] H. Zen, V. Dang, R. Clark, *et al.*, "LibriTTS: A Corpus Derived from LibriSpeech for Text-to-Speech," in *Proc. Interspeech*, 2019, pp. 1526–1530.

[4] W. N. Campbell, "Synthesizing spontaneous speech," in *Computing Prosody: Computational Models for Processing Spontaneous Speech*, Y. Sagisaka, N. Campbell, and N. Higuchi, Eds., New York, NY: Springer US, 1997, pp. 165–186, ISBN: 978-1-4612-2258-3. DOI: 10.1007/978-1-4612-2258-3_12.

[5] M. Wester, O. Watts, and G. E. Henter, "Evaluating comprehension of natural and synthetic conversational speech," in *Proc. Speech Prosody*, 2016, pp. 766–770.

[6] E. Rodero, R. F. Potter, and P. Prieto, "Pitch range variations improve cognitive processing of audio messages," *Human Communication Research*, vol. 43, no. 3, pp. 397–413, 2017.

[7] É. Székely, G. E. Henter, J. Beskow, and J. Gustafson, "Breathing and Speech Planning in Spontaneous Speech Synthesis," in *Proc. ICASSP*, 2020, pp. 7649–7653.

[8] É. Székely, J. Mendelson, and J. Gustafson, "Synthesising Uncertainty: The Interplay of Vocal Effort and Hesitation Disfluencies," in *Proc. Interspeech*, 2017, pp. 804–808.
[9] H. H. Clark and T. Wasow, "Repeating words in spontaneous speech," *Cognitive psychology*, vol. 37, no. 3, pp. 201–242, 1998.
[10] H. H. Clark and J. E. F. Tree, "Using uh and um in spontaneous speaking," *Cognition*, vol. 84, no. 1, pp. 73–111, 2002.
[11] M. Watanabe, K. Hirose, Y. Den, and N. Minematsu, "Filled pauses as cues to the complexity of upcoming phrases for native and non-native listeners," *Speech Communication*, vol. 50, no. 2, pp. 81–94, 2008, ISSN: 0167-6393.
[12] S. H. Fraundorf and D. G. Watson, "The disfluent discourse: Effects of filled pauses on recall," *Journal of Memory and Language*, vol. 65, no. 2, pp. 161–175, 2011, ISSN: 0749-596X.
[13] É. Székely, G. Eje Henter, J. Beskow, and J. Gustafson, "How to train your fillers: Uh and um in spontaneous speech synthesis," in *Proc. 10th ISCA Workshop on Speech Synthesis (SSW 10)*, 2019, pp. 245–250.
[14] É. Székely, G. E. Henter, J. Beskow, and J. Gustafson, "Spontaneous conversational speech synthesis from found data," in *Proc. Interspeech*, 2019, pp. 4435–4439.
[15] S. Wang, J. Gustafson, and É. Székely, "Evaluating Sampling-based Filler Insertion with Spontaneous TTS," in *Proc. LREC*, N. Calzolari, F. Béchet, P. Blache, *et al.*, Eds., Marseille, France, Jun. 2022, pp. 1960–1969.
[16] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc ICLR*, Y. Bengio and Y. LeCun, Eds., 2014.
[17] D. Yoshioka, Y. Yasuda, and T. Toda, "Nonparallel Spoken-Text-Style Transfer for Linguistic Expression Control in Speech Generation," *IEEE/ACM TASLP*, vol. 33, pp. 333–346, 2025.
[18] T. Miyake, "Grammaticalization in modern japanese : On the continuum between content words and function words," *Studies in the Japanese Language*, vol. 1, no. 3, pp. 61–76, 2005. DOI: 10.20666/nihongonokenkyu.1.3_61.
[19] J. Gustafson, J. Beskow, and É. Székely, "Personality in the mix - investigating the contribution of fillers and speaking style to the perception of spontaneous speech synthesis," in *Proc. SSW*, G. Németh, Ed., 2021, pp. 48–53.
[20] K. Maekawa, H. Koiso, S. Furui, and H. Isahara, "Spontaneous Speech Corpus of Japanese," in *Proc. LREC*, 2000, pp. 947–952.
[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[22] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Proc. NeurIPS*, I. Guyon, U. von Luxburg, S. Bengio, *et al.*, Eds., 2017, pp. 5998–6008.
[23] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Association for Computational Linguistics, 2019, pp. 4171–4186.
[24] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. EMNLP*, 2014, pp. 1746–1751.
[25] K. Papineni, S. Roukos, T. Ward, and W. Zhu, "BLEU: a Method for Automatic Evaluation of Machine Translation," in *Proc. ACL*, 2002, pp. 311–318.
[26] S. Lloyd, "Least squares quantization in PCM," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.
[27] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008.