# RWEN-TTS: Relation-Aware Word Encoding Network for Natural Text-to-Speech Synthesis

**Shinhyeok Oh***, **HyeongRae Noh***, **Yoonseok Hong, and Insoo Oh**

Netmarble AI Center
{kai, hr_noh, yhong, ioh}@netmarble.com

## Abstract

With the advent of deep learning, a huge number of text-to-speech (TTS) models which produce human-like speech have emerged. Recently, by introducing syntactic and semantic information w.r.t the input text, various approaches have been proposed to enrich the naturalness and expressiveness of TTS models. Although these strategies showed impressive results, they still have some limitations in utilizing language information. First, most approaches only use graph networks to utilize syntactic and semantic information without considering linguistic features. Second, most previous works do not explicitly consider adjacent words when encoding syntactic and semantic information, even though it is obvious that adjacent words are usually meaningful when encoding the current word. To address these issues, we propose Relation-aware Word Encoding Network (RWEN), which effectively allows syntactic and semantic information based on two modules (i.e., Semantic-level Relation Encoding and Adjacent Word Relation Encoding). Experimental results show substantial improvements compared to previous works.

## Introduction

Text-to-Speech (TTS), which aims at synthesizing natural-sounding speech from text, has extensive applications in various industries such as entertainment, education, and so on (Tan et al. 2021). Recently, deep learning-based TTS models have drawn attention, showing unprecedented results. Most existing works have adopted a two-stage generation scheme, which produces an intermediate speech representation (e.g., Mel-spectrogram) from the input text and then generates a raw waveform. In this work, we focus on the model used in the first stage, called an acoustic model. Generally, the acoustic model is categorized into the autoregressive (AR) model and the non-autoregressive (NAR) model, according to the generation method. Early studies usually focused on the AR model (van den Oord et al. 2016; Skerry-Ryan et al. 2018; Shen et al. 2018; Li et al. 2019). However, they have a slow inference speed caused by sequential generation. Moreover, they are quite sensitive to the alignment resulting in low robustness (e.g., long pause, word repeating, and word skipping). To overcome these limitations, many NAR models (Łańcucki 2021; Ren et al. 2019, 2021) have been proposed. Compared to AR models, they showed faster inference speed by generating speech in parallel and alleviated robustness issues. Nevertheless, their quality of expressiveness is unsatisfactory because they predict prosodic features that contain pitch, duration, and energy without introducing dependency between time steps (Kharitonov et al. 2022). Thus, various approaches to improve the quality of NAR-TTS have been proposed. Min et al. (2021) successfully achieved expressive speech synthesis by introducing a reference encoder that models desired prosody because the same sentence can be uttered in diverse styles. Hwang et al. (2021); Song et al. (2022); Lajszczak et al. (2022) claimed that the performance of NAR-TTS is poor when the training data is insufficient, devising effective data augmentation methods. Kim, Kong, and Son (2021) combined powerful generative models (i.e., variational autoencoder, normalizing flow, and generative adversarial network) to improve expressiveness. They reported that proposed model close to human-level speech. Meanwhile, Kenter, Sharma, and

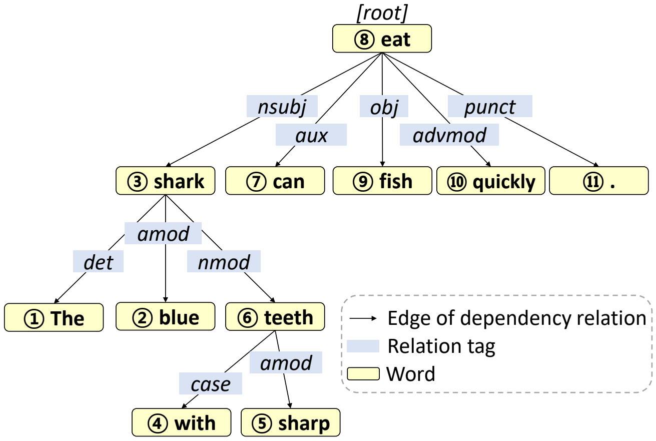

Figure 1: An example of a dependency tree to illustrate for "The blue shark with sharp teeth can eat fish quickly." The description of each element is described in the bottom right corner. For example, "*det*" is the relation tag between "The" and "shark".

[*] These authors contributed equally.

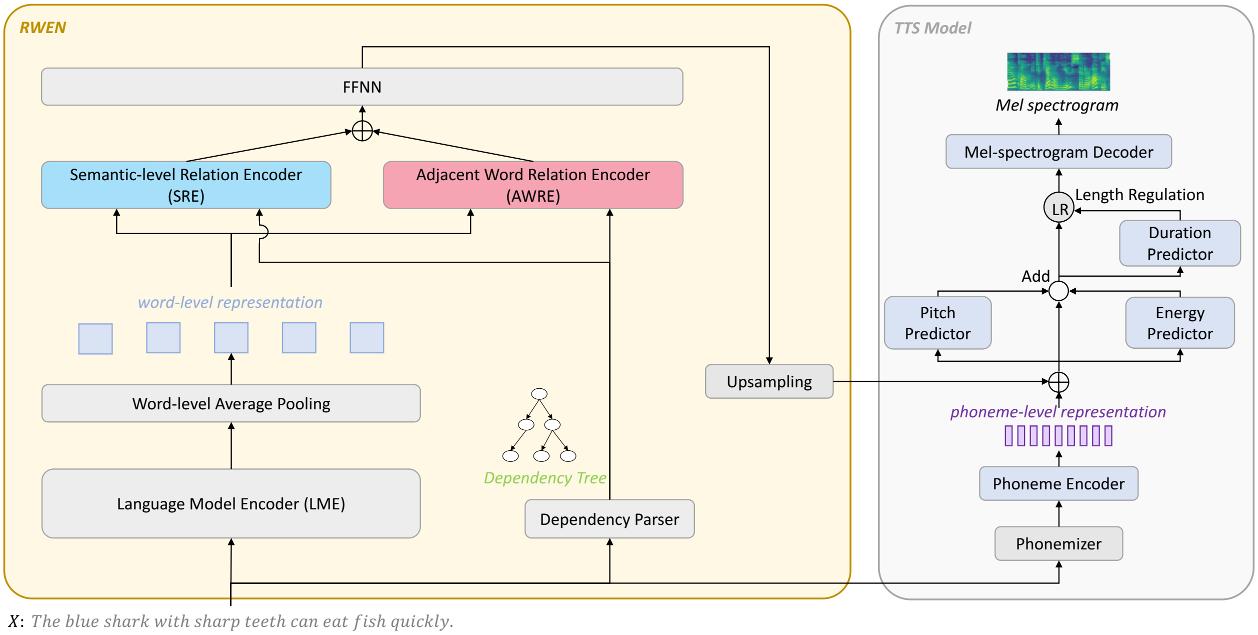
RWEN
FFNN
Semantic-level Relation Encoder (SRE)
Adjacent Word Relation Encoder (AWRE)
word-level representation
Word-level Average Pooling
Language Model Encoder (LME)
Dependency Tree
Dependency Parser
Upsampling
TTS Model
Mel spectrogram
Mel-spectrogram Decoder
LR
Length Regulation
Duration Predictor
Add
Pitch Predictor
Energy Predictor
phoneme-level representation
Phoneme Encoder
Phonemizer
X: The blue shark with sharp teeth can eat fish quickly.

Figure 2: Overall architecture of RWEN for TTS. ⊕ denotes concatenation operator and ◯ denotes element-wise add operator. Length regulation refers to upsampling by repeating for each phonemic representation as much as the predicted duration.

Clark (2020); Liu, Sisman, and Li (2021); Jia et al. (2021); Zhou et al. (2022); Zhang et al. (2022); Tatanov, Beliaev, and Ginsburg (2022) boosted the expressiveness of speech by applying various methods proposed in the field of natural language processing (NLP) to the speech domain. Especially, GraphSpeech (Liu, Sisman, and Li 2021) and Relational Gated Graph Network (RGGN) (Zhou et al. 2022) claimed the syntactic and semantic information of text affects the naturalness and expressiveness of speech. They improved the performance by utilizing graph networks focused on the representation based on dependency relations.

Despite the impressive results, we point out two crucial problems in applying syntactic and semantic information. First, most previous works utilizing dependency relations tend to assign graph networks to encode the neighbor nodes based on the dependency tree. For example, in Figure 1, when encoding "shark", RGGN utilizes weighted-sum to encode "the", "blue", and "teeth", simultaneously. In RGGN, these neighbor words are explicitly considered, and others are implicitly considered. However, "blue" and "teeth" do not have a direct semantic correlation, except they share the same parent. We assume that encoding dimly correlated words simultaneously and explicitly can confuse the model. Second, previous works do not explicitly consider dependency relations on adjacent words. On the other hand, it is obvious the relations of adjacent words are usually meaningful because the TTS task deals with sequential data.

To address the aforementioned issues, we propose Relation-aware Word Encoding Network (RWEN) for TTS. RWEN, which consists of Semantic-level Relation Encoding (SRE) and Adjacent Word Relation Encoding (AWRE), focuses on effectively encoding dependency relations to improve naturalness and expressiveness. SRE encodes dependency relations based on the semantic level to substitute the inefficient graph networks mentioned above. AWRE explicitly encodes dependency relations based on adjacent words. We briefly summarize our main contributions as follows:

- We design two novel approaches, SRE and AWRE, to consider linguistic features and TTS characteristics.
- We propose RWEN that contains SRE and AWRE, which can be easily incorporated into most recent TTS models.
- Experimental results demonstrate that RWEN outperforms existing works, and we prove that SRE and AWRE are significantly effective through our ablation experiments.

# Proposed Method

## RWEN: Relation-aware Word Encoding Network

Figure 2 describes the overall architecture of RWEN. As mentioned before, to solve two crucial problems when assigning dependency relations, we propose a novel approach called RWEN that contains SRE and AWRE.

**Task Description** Given a text $X$, we aim to generate the natural and expressive speech.

**Dependency Parser** We utilize a dependency parser to get the dependency tree. The tree has dependency relations between words, as shown in Figure 1. The dependency parser takes an input sequence represented as,

$$\mathbf{X}^W = [X_1^W \; X_2^W \; ... \; X_n^W], \qquad (1)$$

where $X^W$ represents the list of tokens divided on the basis of words and $n$ denotes the number of words. And the output is a tree like the one described in Figure 1. The output tree consists of heads and relation tags represented as, $head = [head_1, head_2, ..., head_n]$ and $rel = [rel_1, rel_2, ..., rel_n]$. To utilize dependency relations, we define the Relation Tag Embedding (RTE), which makes embeddings for each relation tag,

$$\begin{aligned} E_i^T &= \Phi_{RTE}(rel_i) \\ E^T &= [E_1^T, E_2^T, ..., E_i^T, ..., E_n^T], \end{aligned} \tag{2}$$

where $i$ is an index in the range of $n$ and $\Phi_{RTE}$ denotes the embedding look-up table for RTE. $E^T \in R^{d_{E^T} \times n}$ is embedded representations for one sentence, where $d_{E^T}$ represents a dimension of $\Phi_{RTE}$. Finally, $E^T$ is fed into SRE and AWRE.

**Language Model Encoder** Following recent works, we utilize pre-trained language models, such as BERT (Devlin et al. 2019) and ELECTRA (Clark et al. 2020), as Language Model Encoder (LME) described in Figure 2 to construct text representation. The input sequence for LME is represented as,

$$\mathbf{X}^S = [[\text{CLS}]\, X_1^S\, X_2^S\, ...\, X_m^S\, [\text{SEP}]], \tag{3}$$

where $X^S$ represents the list of tokens divided on the basis of subwords and $m$ denotes the length of tokens for the input sentence tokenized by the pre-trained language model tokenizer. $X^S$ is fed into the pre-trained language model to obtain the output text representation, $H^S = [H^S_{[\text{CLS}]}, H^S_1, H^S_2, ..., H^S_m, H^S_{[\text{SEP}]}] \in R^{d_H \times (m+2)}$, where $d_H$ represents a dimension of the pre-trained language model.

**Word-level Average Pooling** While the dependency relations are divided based on words, $H^S$ are split based on subwords. We need the proper way to align the dependency relations with $H^S$ because we use them simultaneously. Therefore, we utilize Word-level Average Pooling to align between them, similar to Subword-to-Word Mapping by Zhou et al. (2022). We use average pooling ($AP$) based on word level represented as,

$$\begin{aligned} H_i^W &= AP([H_j^S, H_{j+1}^S, ..., H_z^S]) \\ H^W &= [H^W_{[\text{CLS}]}, H_1^W, ..., H_i^W, ..., H_n^W, H^W_{[\text{SEP}]}], \end{aligned} \tag{4}$$

where $j$ and $z$ are the start and end index on subword-level based on $X_i^W$, respectively. For example, if the word "quickly" is tokenized as "quick" and "ly", $H_i^W$ can be represented as $H_i^W = AP([H^S_{quick}, H^S_{ly}])$, where $H^S_{quick}$ and $H^S_{ly}$ denote output text representations for indexes of "quick" and "ly" in $H^S$. $H^W \in R^{d_H \times (n+2)}$ represents the word-level representation described in Figure 2. $H^W_{[\text{CLS}]}$ and $H^W_{[\text{SEP}]}$ in $H^W$ are equal to $H^S_{[\text{CLS}]}$ and $H^S_{[\text{SEP}]}$, respectively. Finally, $H^W$ is fed into SRE and AWRE.

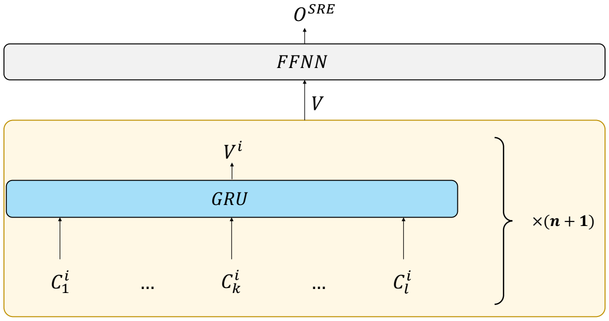

Figure 3: The architecture of SRE. Each GRU encodes vectors concatenated with the word-level representation and RTE.

**Semantic-level Relation Encoding** Previous work (Zhou et al. 2022) proposes RGGN utilizing the graph networks based on the dependency tree. Since the feature of the dependency tree is related to the prosody of speech (Köhn, Baumann, and Dörfler 2018; Liu, Sisman, and Li 2021; Zhou et al. 2022) and graph networks are suitable for encoding tree structures, they report improved results compared to baseline. However, they only use graph networks to encode neighbor words in the tree and don't seem to consider linguistic features. This method can encode the tree structure, but it is inefficient because linguistic features are not considered. Therefore, we propose SRE to effectively encode phrases with contextual meaning. We assume that the phrase from each word to the root can be defined as phrases with their contextual meaning because they are sequentially connected in a dependency tree. SRE is described in Figure 3 and as follows.

SRE aims to encode phrases from each word to the root. Each word has a phrase with contextual meaning, and the indexes from the word to the root node are represented as,

$$I^i = [I_1^i, ..., I_l^i], \tag{5}$$

where $l$ is the length from the current node $i$ to the root node. $I^i$ denotes word indexes in the phrase starting with the index $i$ of each word. For example in Figure 1, if the value of $i$ is 2, $I^2$ is represented as, $I^2 = \{2, 3, 8\}$. To expand to the sentence, the indexes can be represented as,

$$I = [I^0, I^1, I^2, ..., I^n, I^{n+1}], \tag{6}$$

where $I^0$ is the index of $H^W_{[\text{CLS}]}$ and $I^{n+1}$ is the index of $H^W_{[\text{SEP}]}$. Then, we utilize the word-level representation and RTE as follows:

$$\begin{aligned} C_k^i &= H^W_{I_k^i} \oplus E^T_{I_k^i} \\ C^i &= [C_1^i, C_2^i, ..., C_k^i, ..., C_l^i], \end{aligned} \tag{7}$$

where $k$ is an index of $I^i$, and $\oplus$ denotes the concatenation operator. $C^i$ represents a vector with the contextual meaning and dependency relations. $C^i$ is fed into Gated Recurrent Units (GRU) (Chung et al. 2014), and the output is represented as,

$$\begin{aligned} V^i &= GRU(C^i) \\ V &= [V^0, V^1, ..., V^{n+1}], \end{aligned} \tag{8}$$

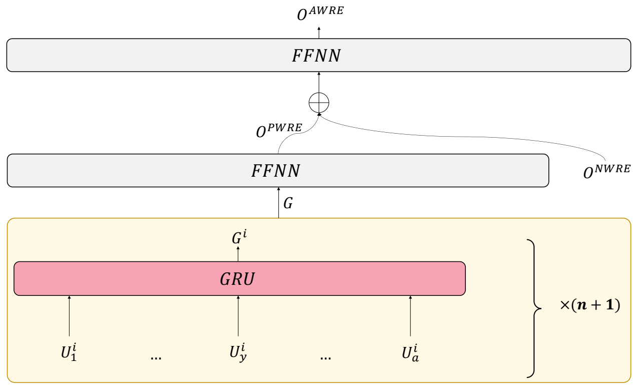

Figure 4: The architecture of AWRE. Each GRU encodes vectors concatenated with the word-level representation, RTE, and DE.

where $V^i$ is the last hidden state of GRU. And, $V$ is fed into single-layer feed-forward neural network (FFNN) as,

$$O^{SRE} = \omega_1 V + b_1, \tag{9}$$

where $\omega_1 \in R^{d_H \times (d_H + d_{ET})}$ and $b_1$ are trainable parameters.

**Adjacent Word Relation Encoding** To improve the results of TTS, we assign TTS characteristics as well as linguistic features. In particular, we focus that both input and output of TTS are sequential data, which are affected by surrounding words. To consider this, we propose AWRE, which encodes dependency relations between surrounding words and the current word based on the dependency tree. AWRE consists of two modules: Previous Word Relation Encoding (PWRE) and Next Word Relation Encoding (NWRE). PWRE encodes the dependency relations from the current word to the previous word, and NWRE encodes the dependency relations from the current word to the next word. First, PWRE is described in Figure 4 and as follows.

PWRE aims to encode dependency relations from the current word to the previous word. We construct the shortest path from the current word to the previous word in the dependency tree. Indexes in the shortest path are represented as,

$$P^i = [P^i_1, ..., P^i_y, ..., P^i_a] \tag{10}$$

where $a$ is the length of the shortest path from the current node to the previous node. $P^i$ denotes indexes starting from the current node $i$ and ending with the previous node $i-1$. For example in Figure 1, if the value of $i$ is 2, $P^2$ is represented as, $P^2 = \{2, 3, 1\}$. To expand to the sentence, the indexes can be represented as,

$$P = [P^0, ..., P^i, ..., P^n, P^{n+1}], \tag{11}$$

where $P^0$ is the index of $H^W_{[\text{CLS}]}$ and $P^{n+1}$ is the index of $H^W_{[\text{SEP}]}$. Additionally, we utilize directions between connected nodes based on the dependency tree represented as,

$$Q_{\{P^i_{y-1}, P^i_y\}} \in [self, parent, child], \tag{12}$$

where $Q_{\{P^i_{y-1}, P^i_y\}}$ denotes the direction between $P^i_{y-1}$ and $P^i_y$. If the direction from $P^i_{y-1}$ to $P^i_y$ is the parent, $Q_{\{P^i_{y-1}, P^i_y\}}$ denotes $parent$. Likewise, if the direction is the child, $Q_{\{P^i_{y-1}, P^i_y\}}$ denotes $child$. When it needs to encode itself (i.e., first element), we use $self$ to utilize the direction. Then, we define Direction Embedding (DE) represented as,

$$\begin{aligned} D^i = &\{\Phi_{DE}(self), ..., \Phi_{DE}(Q_{\{P^i_{y-1}, P^i_y\}}) \\ &, ..., \Phi_{DE}(Q_{\{P^i_{a-1}, P^i_a\}}))\}, \end{aligned} \tag{13}$$

where $\Phi_{DE} \in R^{d_{\Phi_{DE}} \times 3}$ denotes the embedding look-up table for DE. $R^{d_{\Phi_{DE}}}$ represents a dimension of $\Phi_{DE}$. DE is only used in AWRE. In SRE, we do not consider to use DE because it is encoded only in one direction based on the tree.

We utilize the word-level representation, RTE and DE as follows:

$$\begin{aligned} U^i_y &= H^W_{P^i_y} \oplus E^T_{P^i_y} \oplus D^i_{\{P^i_{y-1}, P^i_y\}} \\ U^i &= [U^i_1, U^i_2, ..., U^i_y, ..., U^i_a], \end{aligned} \tag{14}$$

where $U^i$ represents a vector with the contextual meaning, dependency relations, and the embedding of directions. $U^i$ is fed into GRU, and the output is represented as,

$$\begin{aligned} G^i &= GRU(U^i) \\ G &= [G^0, G^1, ..., G^{n+1}], \end{aligned} \tag{15}$$

where $G^i$ is the last hidden state of GRU. And, $G$ is fed into FFNN as,

$$O^{PWRE} = \omega_2 G + b_2, \tag{16}$$

where $\omega_2 \in R^{d_H \times (d_H + d_{ET} + d_{\Phi_{DE}})}$ and $b_2$ are trainable parameters.

NWRE is encoded similarly to PWRE, which has Equations 10, 11, 12, 14, 15, and 16. NWRE can be described by replacing the previous node only with the next node in the PWRE description. Thus, note that the final output of NWRE is represented $O^{NWRE}$. Then, $O^{PWRE}$ and $O^{NWRE}$ are concatenated as,

$$O^{P\&N} = O^{PWRE} \oplus O^{NWRE}, \tag{17}$$

and fed into FFNN as,

$$O^{AWRE} = \omega_3 O^{P\&N} + b_3, \tag{18}$$

where $\omega_3 \in R^{d_H \times (d_H + d_H)}$ and $b_3$ are trainable parameters.

**Upsampling** $O^{SRE}$ and $O^{AWRE}$ are concatenated and fed into FFNN. Then, the output is represented by the word-level. We should concatenate with the output and phoneme-level representation, as shown in Figure 2. However, phoneme-level representation is represented by the phoneme-level so that we can't directly concatenate. Thus, an upsampling method is required to concatenate with them. We duplicate the word-level segmented output representation by the number of phoneme sequences corresponding to each word and concatenate it with the phoneme-level representation.

### TTS Model

To prove the effectiveness of our method, we adapt FastPitch (Łańcucki 2021) equipped with Unsupervised Alignment Learning framework (UAL) (Badlani et al. 2022) as the TTS model, which is one of the representative NAR-TTS models. More specifically, as shown in Figure 2, it consists of five modules: Phoneme Encoder, Mel-spectrogram Decoder, Pitch Predictor, Energy Predictor, and Duration Predictor. Phoneme Encoder produces the phoneme-level representation from the phonemic text. Then, Pitch Predictor and Energy Predictor take the phoneme-level representation concatenated with the output representation of RWEN, constructing the pitch and energy information. With the help of UAL, Duration Predictor can be learned to predict the duration of each phoneme, which is used to perform upsampling from phoneme-level representation to frame-level one. Note that the prosody of synthesized speech can be controlled by adjusting the predicted pitch and duration during the inference stage. Finally, Mel-spectrogram Decoder generates the output Mel-spectrogram from the frame-level representation.

## Experiments

### Experimental Setup

**Datasets** We train and evaluate RWEN on LJSpeech (Ito and Johnson 2017), a single speaker corpus recorded by a female English speaker. It consists of 13,100 short audio clips with a total length of 24 hours, being randomly split into 12,500, 100, and 500 samples to comprise the training, validation, and test datasets as in Kim, Kong, and Son (2021). Additionally, recent works (Kim, Kong, and Son 2021; Tan et al. 2022) have already achieved human-level performance on the benchmark datasets (e.g., LJSpeech, VCTK (Yamagishi et al. 2019), etc.). Therefore, we evaluate RWEN on other type of datasets used in the field of NLP in order to derive meaningful comparison results. To cover multiple domains, we evaluate RWEN on the following datasets:

- **CNN/Daily Mail** (Nallapati et al. 2016) contains articles from CNN and DailyMail newspapers.
- **Children's Book Test (CBT)** (Hill et al. 2016) contains sentences built from books for children that are freely available.
- **OpenBookQA** (Mihaylov et al. 2018) contains a small book of core elementary-level science facts.
- **SQuAD 2.0** (Rajpurkar, Jia, and Liang 2018) contains sentences on a set of Wikipedia articles.

**Subjective Evaluation** We conducted the crowd-sourced listening test for Mean Opinion Score (MOS) and Comparative Mean Opinion Score (CMOS) on Amazon Mechanical Turk [1]. We used at least 50 sentences randomly sampled from each dataset for all evaluations, and at least 15 listeners participated. To maintain evaluation quality, master workers certificated in Amazon Mechanical Turk only participated, and all submissions of workers who did not pass occasional hearing tests were rejected. For MOS, we evaluated naturalness and expressiveness on a 5-point scale from 1 to 5. For CMOS, we evaluated which one is more natural and more expressive on a 7-point scale from -3 to 3. Also, CMOS was measured between the baseline and a comparative model. Therefore, it is only possible to compare models between the baseline and a comparative model.

| | MOS (CI) |
|---|---|
| VITS (Kim, Kong, and Son 2021) | 3.95 (±0.06) |
| FastPitch (Łańcucki 2021) | 3.74 (±0.06) |
| FastPitch w/ UAL | 4.04 (±0.06) |

Table 1: Evaluation results for existing TTS models on the CNN/Daily Mail dataset. We measured with Mean Opinion Score (MOS) and 95% confidence intervals (CI).

**Baselines** For experiments, we compare our model with followings:

- **VITS** (Kim, Kong, and Son 2021) is a fully end-to-end TTS model that produces human-like sounding audio on the waveform domain by leveraging variational autoencoder (VAE) (Kingma and Welling 2014) with normalizing flows and adversarial training. To solve the one-to-many problem that one text can be spoken in various styles, they also introduced a flow-based stochastic duration predictor, demonstrating significant effectiveness. In this work, we used the official pre-trained model for fair comparisons [2].
- **FastPitch** (Łańcucki 2021) is an acoustic model which generates a Mel-spectrogram from given text. It can control the pitch and duration of the synthesized speech by adjusting the outputs of pitch and duration predictors. In this work, we used the official checkpoint [3] for fair comparisons. And, since it generates a Mel-spectrogram from text, we need a model called vocoder that converts a Mel-spectrogram into a raw waveform. To this end, we use the official HifiGAN (Kong, Kim, and Bae 2020) codes [4] and a checkpoint pre-trained on the LJSpeech dataset and finetuned as the output of Tacotron 2 (Shen et al. 2018).
- **FastPitch w/ UAL** is a TTS model that contains FastPitch and Unsupervised Alignment Learning framework. We constructed by referring to codes of the official FastPitch repository [5]. In addition, we modified the source code so that it can be processed in phoneme-level sequences for fair comparisons with our proposed model and RGGN.
- **RGGN-BERT** (Zhou et al. 2022) proposed RGGN to improve the naturalness and expressiveness of synthe-

[1] https://www.mturk.com/

[2] https://github.com/jaywalnut310/vits

[3] https://github.com/NVIDIA/DeepLearningExamples/blob/8d8c524df634e4dfa0cfbf77a904ce2ede85e2ec/PyTorch/SpeechSynthesis/FastPitch/scripts/download_fastpitch.sh

[4] https://github.com/jik876/hifi-gan

[5] https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/SpeechSynthesis/FastPitch

| | MOS (CI) | | | | |
|---|---|---|---|---|---|
| | LJSpeech | News | Book | | Wiki |
| | | CNN/Daily Mail | CBT | OpenBookQA | SQuAD 2.0 |
| Ground Truth | 4.25 (± 0.06) | - | - | - | - |
| VITS | 4.04 (± 0.06) | 4.01 (± 0.06) | 3.94 (± 0.05) | 3.98 (± 0.06) | 4.03 (± 0.06) |
| FastPitch w/ UAL | 4.16 (± 0.06) | 4.05 (± 0.06) | 4.06 (± 0.05) | 3.91 (± 0.07) | 4.10 (± 0.06) |
| RGGN-BERT | 4.15 (± 0.06) | 4.00 (± 0.06) | 4.07 (± 0.05) | 3.95 (± 0.07) | 4.12 (± 0.06) |
| RWEN-BERT | **4.19 (± 0.06)** | **4.15 (± 0.06)** | **4.15 (± 0.05)** | **4.00 (± 0.06)** | **4.18 (± 0.06)** |

Table 2: Evaluation results of MOS with 95% CI. The best scores except Ground Truth are in bold. '-' denotes the dataset doesn't have voices of Ground Truth or can't be evaluated because speakers are different between the training dataset and evaluation dataset. MOS was measured simultaneously within each column so that it is possible to compare models within the same column.

sized speeches. They utilized dependency structure and pre-trained BERT (Devlin et al. 2019) embedding. For their experiments, they used Tacotron 2 (Shen et al. 2018) as the TTS model. However, in this work, we implemented RGGN with FastPitch w/ UAL for fair comparisons.

**TTS system for RWEN** To prove the effectiveness of our proposed method, we adopt FastPitch w/ UAL. Compared to the recent end-to-end TTS model (e.g., VITS), FastPitch w/ UAL has controllability in terms of pitch and duration, which can utilize various applications. Also, it is light and easy to conduct diverse experiments. Moreover, as shown in Table 1, FastPitch w/ UAL achieved the best performance by introducing UAL and phoneme-level encoding.

## Implementation Details

We implemented our proposed model, called RWEN, using the PyTorch (Paszke et al. 2019) and Transformers[6] (Wolf et al. 2020) library. We adopt BERT$_{base}$ and ELECTRA$_{base}$ (Clark et al. 2020) as the LME for our experiments. In our experiments, RWEN-BERT and RWEN-ELECTRA denotes RWEN using BERT$_{base}$ and ELECTRA$_{base}$ as the LME, respectively. Following RGGN, we use Stanza (Qi et al. 2020) to get the dependency tree. We use FastPitch w/ UAL as our TTS model to simplify the experiments. Specifically, Phoneme Encoder and Mel-spectrogram Decoder are composed of four Feed-Forward Transformer (FFT) blocks (Ren et al. 2019) whose parameters are the same as described in Ren et al. (2021) except that the hidden size of the Mel-spectrogram decoder is 1024. Duration Predictor, Pitch Predictor, and Energy Predictor are the same architecture: two 1-D convolutions with kernel size 3 and 256/256 input/output channels, each followed by ReLU, LayerNorm, and Dropout with the probability of 0.1. To extract the target pitch from the speech, we use the pYIN (Mauch and Dixon 2014) algorithm and perform normalization with the mean and standard deviation of the pitch for the whole training dataset. Also, the energy of speech is extracted by performing the L2 norm on the Mel-spectrogram. The last fully connected layer projects a 256-dimensional vector into a scalar. To produce raw waveform from the synthesized Mel-spectrogram, pre-trained HiFiGAN (Kong, Kim, and Bae 2020) is used as the vocoder. Besides, we utilized the phonemizer (Bernard and Titeux 2021) since we used the phoneme sequence as the input. We use mixed precision training on 16 Tesla A100 GPUs for all the experiments. The batch size is set to 2 per GPU, and the model is trained up to 200k steps. More details and samples are in our repository[7] and demonstration site[8].

| | CMOS | Wilcoxon p-value |
|---|---|---|
| RWEN-BERT | 0 | - |
| w/o AWRE | $-0.09$ | 7.5e-5 |
| w/o SRE | $-0.11$ | 2.6e-6 |
| w/o SRE & AWRE | $-0.13$ | 4.1e-7 |

Table 3: Ablation study on the CNN/Daily Mail dataset. We choose RWEN-BERT as the baseline. We measured with CMOS and Wilcoxon p-value obtained by Wilcoxon signed rank test (Wilcoxon 1992).

## Overall Results

Table 2 reports MOS results to evaluate on LJSpeech, CNN/-Daily Mail, CBT, OpenBookQA, and SQuAD 2.0 datasets. We observe that RWEN-BERT shows slightly lower MOS than Ground Truth in the LJSpeech evaluation dataset, and RWEN-BERT outperforms the comparative models for all datasets. This suggests that the proposed approaches are effective for TTS. For all datasets except OpenBookQA, FastPitch w/ UAL shows higher MOS than VITS. This can be additional evidence for Table 1, indicating that the naturalness and expressiveness of FastPitch w/ UAL are similar to or better than VITS. FastPitch w/ UAL and RGGN-BERT show similar performance. As we mentioned in the introduction section, it can be seen that RGGN reflects syntactic and semantic information inefficiently. As a result, RWEN-BERT achieves gains over FastPitch w/ UAL we utilized as our TTS system by 0.03 (4.16 → 4.19), 0.10

[6] https://github.com/huggingface/transformers

[7] https://github.com/shinhyeokoh/rwen

[8] https://shinhyeokoh.github.io/demo/rwen_tts/index.html

|  | CMOS | Wilcoxon p-value |
|---|---|---|
| RWEN-BERT | 0 | - |
| RWEN-ELECTRA | +0.22 | 1.4e-13 |

Table 4: CMOS results on the CNN/Daily Mail dataset. To study effectiveness according to change of PLM, we choose RWEN-BERT as the baseline. We measured with CMOS and Wilcoxon p-value.

(4.05 → 4.15), 0.09 (4.06 → 4.15), 0.09 (3.91 → 4.00), and 0.08 (4.10 → 4.18) on the LJSpeech, CNN/Daily Mail, CBT, OpenBookQA, and SQuAD 2.0 datasets, respectively.

### Ablation Study

To study the effects of AWRE and SRE, we conduct ablation experiments on CNN/Daily Mail dataset. As shown in Table 3, we set the baseline that RWEN-BERT. Removing AWRE (i.e., only utilizing SRE) and SRE (i.e., only utilizing AWRE) brings 0.09 and 0.11 CMOS degradation, respectively. If we remove SRE and AWRE (i.e., utilizing FastPitch w/ UAL), it brings 0.13 CMOS degradation. We can observe CMOS drop significantly in all ablation experiments. This suggests all of our proposed approaches are effective for the TTS model. Meanwhile, we can also observe that the SRE is more effective than the AWRE. As SRE encodes phrases with contextual meaning, it allows the model to more exploit the dependency relations at a sentence-level.

### Effects of Pre-trained Language Model

To study effectiveness according to the change of the pre-trained language model, we conduct CMOS experiments between RWEN-BERT and RWEN-ELECTRA. RWEN-ELECTRA uses ELECTRA$_{base}$ as LME in our architecture. Clark et al. (2020) reports that ELECTRA-base outperforms BERT-base on the GLUE (Wang et al. 2018) widely used benchmark for natural language understanding. As shown in Table 4, RWEN-ELECTRA significantly improves compared to RWEN-BERT. This suggests that the quality of the pre-trained language model affects RWEN, and using the improved language model has a positive effect on our proposed method.

## Conclusion

In this study, we pointed out crucial problems of existing works for TTS that utilize dependency relations based on graph networks. To address these issues, we proposed Relation-aware Word Encoding Network for text-to-speech synthesis. RWEN effectively allows linguistic features to utilize dependency relations and can be easily incorporated into most existing TTS models. Moreover, experimental results show that RWEN outperforms existing works, and we prove that SRE and AWRE are significantly effective through our ablation experiments.

## Acknowledgments

We thank the anonymous reviewers, Kihyuk Jeong, Sunwoo Im for their constructive comments.

## References

Badlani, R.; Łańcucki, A.; Shih, K. J.; Valle, R.; Ping, W.; and Catanzaro, B. 2022. One TTS alignment to rule them all. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6092–6096. IEEE.

Bernard, M.; and Titeux, H. 2021. Phonemizer: Text to Phones Transcription for Multiple Languages in Python. *Journal of Open Source Software*, 6(68): 3958.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Clark, K.; Luong, M.; Le, Q. V.; and Manning, C. D. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Hill, F.; Bordes, A.; Chopra, S.; and Weston, J. 2016. The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Hwang, M.-J.; Yamamoto, R.; Song, E.; and Kim, J.-M. 2021. TTS-by-TTS: TTS-driven data augmentation for fast and high-quality speech synthesis. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6598–6602. IEEE.

Ito, K.; and Johnson, L. 2017. The LJ Speech Dataset. https://keithito.com/LJ-Speech-Dataset/. Accessed: 2022-06-01.

Jia, Y.; Zen, H.; Shen, J.; Zhang, Y.; and Wu, Y. 2021. PnG BERT: Augmented BERT on Phonemes and Graphemes for Neural TTS. In *Interspeech 2021, 22nd Annual Conference of the International Speech Communication Association, Brno, Czechia, 30 August - 3 September 2021*, 151–155. ISCA.

Kenter, T.; Sharma, M.; and Clark, R. 2020. Improving the Prosody of RNN-Based English Text-To-Speech Synthesis by Incorporating a BERT Model. In *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, 4412–4416. ISCA.

Kharitonov, E.; Lee, A.; Polyak, A.; Adi, Y.; Copet, J.; Lakhotia, K.; Nguyen, T. A.; Riviere, M.; Mohamed, A.;

Dupoux, E.; and Hsu, W.-N. 2022. Text-Free Prosody-Aware Generative Spoken Language Modeling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 8666–8681. Dublin, Ireland: Association for Computational Linguistics.

Kim, J.; Kong, J.; and Son, J. 2021. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In *International Conference on Machine Learning*, 5530–5540. PMLR.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Köhn, A.; Baumann, T.; and Dörfler, O. 2018. An Empirical Analysis of the Correlation of Syntax and Prosody. In *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018*, 2157–2161. ISCA.

Kong, J.; Kim, J.; and Bae, J. 2020. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33: 17022–17033.

Lajszczak, M.; Prasad, A.; Van Korlaar, A.; Bollepalli, B.; Bonafonte, A.; Joly, A.; Nicolis, M.; Moinet, A.; Drugman, T.; Wood, T.; et al. 2022. Distribution augmentation for low-resource expressive text-to-speech. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8307–8311. IEEE.

Łańcucki, A. 2021. Fastpitch: Parallel text-to-speech with pitch prediction. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6588–6592. IEEE.

Li, N.; Liu, S.; Liu, Y.; Zhao, S.; and Liu, M. 2019. Neural speech synthesis with transformer network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 6706–6713.

Liu, R.; Sisman, B.; and Li, H. 2021. Graphspeech: Syntax-Aware Graph Attention Network for Neural Speech Synthesis. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6059–6063.

Mauch, M.; and Dixon, S. 2014. PYIN: A fundamental frequency estimator using probabilistic threshold distributions. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*, 659–663. IEEE.

Mihaylov, T.; Clark, P.; Khot, T.; and Sabharwal, A. 2018. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2381–2391. Brussels, Belgium: Association for Computational Linguistics.

Min, D.; Lee, D. B.; Yang, E.; and Hwang, S. J. 2021. Meta-stylespeech: Multi-speaker adaptive text-to-speech generation. In *International Conference on Machine Learning*, 7748–7759. PMLR.

Nallapati, R.; Zhou, B.; dos Santos, C.; Gulçehre, Ç.; and Xiang, B. 2016. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 280–290. Berlin, Germany: Association for Computational Linguistics.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, 8026–8037.

Qi, P.; Zhang, Y.; Zhang, Y.; Bolton, J.; and Manning, C. D. 2020. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 101–108. Online: Association for Computational Linguistics.

Rajpurkar, P.; Jia, R.; and Liang, P. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 784–789. Melbourne, Australia: Association for Computational Linguistics.

Ren, Y.; Hu, C.; Tan, X.; Qin, T.; Zhao, S.; Zhao, Z.; and Liu, T. 2021. FastSpeech 2: Fast and High-Quality End-to-End Text to Speech. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Ren, Y.; Ruan, Y.; Tan, X.; Qin, T.; Zhao, S.; Zhao, Z.; and Liu, T.-Y. 2019. Fastspeech: Fast, robust and controllable text to speech. *Advances in Neural Information Processing Systems*, 32.

Shen, J.; Pang, R.; Weiss, R. J.; Schuster, M.; Jaitly, N.; Yang, Z.; Chen, Z.; Zhang, Y.; Wang, Y.; Ryan, R.; Saurous, R. A.; Agiomyrgiannakis, Y.; and Wu, Y. 2018. Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*, 4779–4783. IEEE.

Skerry-Ryan, R. J.; Battenberg, E.; Xiao, Y.; Wang, Y.; Stanton, D.; Shor, J.; Weiss, R. J.; Clark, R.; and Saurous, R. A. 2018. Towards End-to-End Prosody Transfer for Expressive Speech Synthesis with Tacotron. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, 4700–4709. PMLR.

Song, E.; Yamamoto, R.; Kwon, O.; Song, C.; Hwang, M.; Oh, S.; Yoon, H.; Kim, J.; and Kim, J. 2022. TTS-by-TTS 2: Data-Selective Augmentation for Neural Speech Synthesis Using Ranking Support Vector Machine with Variational Autoencoder. In *Interspeech 2022, 23rd Annual Conference of the International Speech Communication Association, Incheon, Korea, 18-22 September 2022*, 1941–1945. ISCA.

Tan, X.; Chen, J.; Liu, H.; Cong, J.; Zhang, C.; Liu, Y.; Wang, X.; Leng, Y.; Yi, Y.; He, L.; Soong, F. K.; Qin, T.; Zhao, S.; and Liu, T. 2022. NaturalSpeech: End-to-End Text to Speech Synthesis with Human-Level Quality. *CoRR*, abs/2205.04421.

Tan, X.; Qin, T.; Soong, F. K.; and Liu, T. 2021. A Survey on Neural Speech Synthesis. *CoRR*, abs/2106.15561.

Tatanov, O.; Beliaev, S.; and Ginsburg, B. 2022. Mixer-TTS: non-autoregressive, fast and compact text-to-speech model conditioned on language model embeddings. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7482–7486. IEEE.

van den Oord, A.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A. W.; and Kavukcuoglu, K. 2016. WaveNet: A Generative Model for Raw Audio. In *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*, 125. ISCA.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 353–355. Brussels, Belgium: Association for Computational Linguistics.

Wilcoxon, F. 1992. *Individual Comparisons by Ranking Methods*, 196–202. New York, NY: Springer New York. ISBN 978-1-4612-4380-9.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Le Scao, T.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45.

Yamagishi; Junichi, V.; Christophe, M.; and Kirsten. 2019. CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit (version 0.92). https://datashare.ed.ac.uk/handle/10283/3443. Accessed: 2022-06-01.

Zhang, G.; Song, K.; Tan, X.; Tan, D.; Yan, Y.; Liu, Y.; Wang, G.; Zhou, W.; Qin, T.; Lee, T.; and Zhao, S. 2022. Mixed-Phoneme BERT: Improving BERT with Mixed Phoneme and Sup-Phoneme Representations for Text to Speech. In *Interspeech 2022, 23rd Annual Conference of the International Speech Communication Association, Incheon, Korea, 18-22 September 2022*, 456–460. ISCA.

Zhou, Y.; Song, C.; Li, J.; Wu, Z.; Bian, Y.; Su, D.; and Meng, H. 2022. Enhancing Word-Level Semantic Representation via Dependency Structure for Expressive Text-to-Speech Synthesis. In *Interspeech 2022, 23rd Annual Conference of the International Speech Communication Association, Incheon, Korea, 18-22 September 2022*, 5518–5522. ISCA.