SyntaSpeech: Syntax-Aware Generative Adversarial Text-to-Speech

Zhenhui Ye, **Zhou Zhao***, **Yi Ren** and **Fei Wu**
College of Computer Science and Technology, Zhejiang University
{zhenhuiye, zhaozhou, rayeren, wufei}@zju.edu.cn

Abstract

The recent progress in non-autoregressive text-to-speech (NAR-TTS) has made fast and high-quality speech synthesis possible. However, current NAR-TTS models usually use phoneme sequence as input and thus cannot understand the tree-structured syntactic information of the input sequence, which hurts the prosody modeling. To this end, we propose SyntaSpeech, a syntax-aware and light-weight NAR-TTS model, which integrates tree-structured syntactic information into the prosody modeling modules in PortaSpeech. Specifically, 1) We build a syntactic graph based on the dependency tree of the input sentence, then process the text encoding with a syntactic graph encoder to extract the syntactic information. 2) We incorporate the extracted syntactic encoding with PortaSpeech to improve the prosody prediction. 3) We introduce a multi-length discriminator to replace the flow-based post-net in PortaSpeech, which simplifies the training pipeline and improves the inference speed, while keeping the naturalness of the generated audio. Experiments on three datasets not only show that the tree-structured syntactic information grants SyntaSpeech the ability to synthesize better audio with expressive prosody, but also demonstrate the generalization ability of SyntaSpeech to adapt to multiple languages and multi-speaker text-to-speech. Ablation studies demonstrate the necessity of each component in SyntaSpeech. Source code and audio samples are available at https://syntaspeech.github.io.

1 Introduction

Text-to-speech (TTS) aims to synthesize natural speech for input text. Recently, deep learning based TTS has made rapid progress and shown competitive performance with traditional TTS systems [van den Oord *et al.*, 2016]. Neural TTS approaches typically learn an acoustic model that generates the mel-spectrogram or linguistic features from the input sentence [Wang *et al.*, 2017], then adopt a vocoder to synthesize the waveform [van den Oord *et al.*, 2016]. To effectively extract semantic and prosody information from the input text, some previous neural TTS models generate mel-spectrograms autoregressively and suffer from slow inference speed [Ping *et al.*, 2018]. To improve the practicality, non-autoregressive text-to-speech (NAR-TTS) explores to synthesize the mel-spectrogram in parallel [Ren *et al.*, 2019], yet is faced with the difficulty to model expressive prosody using non-autoregressive structures. Recently, NAR-TTS modules tackle this problem by decoupling the prosody into several aspects (such as duration, pitch, etc) [Kim *et al.*, 2020][Ren *et al.*, 2021a], and achieves comparable performance with autoregressive text-to-speech approaches (AR-TTS). Currently, improving the modeling of the prosody is still an open question in NAR-TTS.

Syntactic information, especially the dependency relation, possesses rich intonational features such as pitch accent and phrasing of the input text [Hirschberg and Rambow, 2001]. To be intuitive, we provide an example in Fig.1 to show the potential relationship between the dependency tree and the audio. There are also many TTS extensions utilizing syntactic information to improve prosody. For instance, GraphTTS [Sun *et al.*, 2020] and GraphPB [Sun *et al.*, 2021] construct a syntactic graph based on the character sequence and prosody boundary in the sentence, respectively. GraphSpeech [Liu *et al.*, 2021] and RGNN [Zhou *et al.*, 2021] utilize dependency relation in a sentence and extract the syntactic information with graph neural networks. However, previous syntax-aware TTS models are done in the framework of AR-TTS. Since AR-TTS predicts the duration and pitch autoregressively, it could easily exploit the syntactic information by taking it as auxiliary input features of the backbone. By contrast, NAR-TTS typically models prosody with external predictors, although the extracted features can be used as the auxiliary input features of these prosody predictors, this approach has not been explored yet. To our knowledge, there is no NAR-TTS model that could effectively embed the tree-structured syntactic information to improve the prosody prediction.

To exploit the syntactic information with NAR-TTS, in this work, we propose SyntaSpeech, a syntax-aware generative text-to-speech model, which improves the prosody in the generated mel-spectrogram using a graph encoder to exploit the dependency relation of the raw text, and enhances the audio quality with adversarial training. Specifically,

*Corresponding author

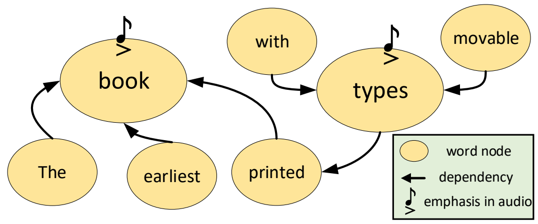


Figure 1: The dependency tree of the input text "*The earliest book printed with movable types*". The emphasis in the real audio is marked with the emphasis symbol.

- To generate the word-level syntactic encoding, we build a syntax graph for each input sentence based on its dependency tree, process the phoneme-level latent encoding to represent the word node in the graph, then aggregate the graphical features with a graph encoder.
- To utilize the extracted syntactic features in prosody modeling, we incorporate the graph encoder into PortaSpeech. The syntactic encoding is embedded into the duration predictor and the variational generator, to improve the duration and pitch prediction, respectively.
- To generate realistic audio with lightweight structures and simplify the training pipeline, we adopt multi-length adversarial training to replace the flow-based post-net in PortaSpeech.

To demonstrate the generalization ability of our SyntaSpeech, we perform experiments on three datasets, including one single-speaker English dataset, one single-speaker Chinese corpus, and one multi-speaker English dataset. Experiments on all datasets show that SyntaSpeech outperforms other state-of-the-art TTS models in voice quality and (especially) prosody in terms of subjective and objective evaluation metrics. The rest of the paper is organized as follows: In Sec.2 we discuss recent progress in NAR-TTS and previous works that develop a syntax-aware TTS model. In Sec.3 we introduce our SyntaSpeech in details. Performance evaluation and ablation studies of SyntaSpeech are given in Sec.4. Finally, we draw conclusions in Sec.5.

2 Related Works

2.1 Non-Autoregressive Text-to-Speech

In the past few years, modern neural TTS thrived with the development of deep learning. Originally, to model the long-term relationships among the input tokens, previous works tend to generate the mel-spectrogram autoregressively [Wang *et al.*, 2017][Ping *et al.*, 2018]. However, AR-TTS is faced with the challenges of slow inference and robustness issues (e.g., word skipping) incurred by autoregressive generation.

To tackle these issues, many works explore adopting non-autoregressive generation. Some works use positional attention for the text and speech alignment[Peng *et al.*, 2020], while the other works use duration prediction to handle the length mismatch between text and mel-frame sequences. For instance, FastSpeech [Ren *et al.*, 2019], Glow-TTS [Kim *et al.*, 2020], and EATS [Donahue *et al.*, 2021] use duration predictor to upsample the phoneme sequence to match the length of mel-spectrograms. These works enjoy fast inference and well robustness. Recent works further improve the expressiveness in NAR-TTS by modeling the variation information. For instance, FastSpeech 2 [Ren *et al.*, 2021a] introduced a pitch predictor to infer the pitch contour in the generated mel-spectrogram. VITS [Kim *et al.*, 2021] and PortaSpeech [Ren *et al.*, 2021b] leverage variational auto-encoder (VAE) to model the variation information in the latent space. To date, improving the expressiveness of the generated waveform is still an open question to the TTS community.

2.2 Syntax-aware Text-to-Speech

Syntax information, which records the dependency relation between the tokens in the text, is acknowledged as a helpful feature to estimate the prosody of the speech and has been studied in speech synthesis before the neural TTS age [Hirschberg and Rambow, 2001][Mishra *et al.*, 2015].

Modern TTS typically utilizes the syntactic information as auxiliary features in AR-TTS modules: GraphTTS [Sun *et al.*, 2020] designs a character-level text-to-graph module to extract the sequential information in the sentence and tries several graph neural networks (GNNs) to process the graphical features. The extracted syntactic feature is then fed into the decoder of Tacotron [Wang *et al.*, 2017] as an auxiliary encoding. Later, GraphSpeech [Liu *et al.*, 2021] introduces *dependency parsing* in the text-to-graph module to better represent the syntactic information of the input sentence, and utilizes bi-directional gated recurrent unit (GRU) to aggregate information through the syntactic graph. Recently, RGGN [Zhou *et al.*, 2021] also adopts dependency parsing to construct the syntactic graph and utilize pre-trained word embedding from BERT [Devlin *et al.*, 2019], then process the graphical data with gated graph neural network (GGNN) [Li *et al.*, 2016]. Both of GraphSpeech and RGGN regard the dependency-based syntactic encoding as auxiliary features and feed them into the encoder of the sequence-to-sequence (seq-to-seq) AR-TTS module.

The difference between SyntaSpeech and previous works is as follows. Firstly, to our knowledge, our SyntaSpeech is the first work that analyzes the syntactic information in NAR-TTS. Secondly, previous works extract syntactic information to provide a better text representation for the seq-to-seq model, while we learn the syntactic encoding for the duration and other prosody attributes prediction, which could make full use of the syntactic features and is more interpretable. Thirdly, previous works either use pre-trained embedding or learn character-level embedding as the node representation in the syntactic graph, by contrast, we process the latent features in the backbone of the TTS model with word-level pooling [Ren *et al.*, 2021b] to formulate the node embedding.

3 SyntaSpeech

To exploit the syntactic information of the input text in the framework of NAR-TTS, we propose SyntaSpeech, which exploits the dependency relation to improve the naturalness and expressiveness of the synthesized audio waveform. In this section, we first introduce a *syntactic graph builder* to construct a syntactic graph based on the input text, which can be utilized in either English or Chinese. Then we design the overall network structure of SyntaSpeech based on PortaSpeech [Ren *et al.*, 2021b]. As shown in Figure 1a, SyntaSpeech designs a *syntactic graph encoder* to provide syntactic information for duration prediction (in linguistic encoder) and other prosody attributes distribution modeling (in variational generator). In general, SyntaSpeech exploits the syntactic information in the raw text with the following steps:

- Firstly, the text sequence is fed into the transformer-based phoneme encoder to obtain the phoneme encoding, which is then processed into a word-level representation with average pooling based on the word boundary.
- Secondly, the syntactic graph builder constructs the syntactic graph using dependency relation, and the word encoding is aggregated through the constructed graph using *gated graph convolution* [Li *et al.*, 2016].
- Thirdly, the obtained word-level syntactic encoding is expanded into phoneme level and frame level, to embed syntactic information into the duration prediction and pitch-energy prediction, respectively.

Besides, we also replace the post-net in PortaSpeech with adversarial training to simplify the training pipeline while keeping the naturalness of the generated mel-spectrogram. We describe these designs in detail in the following subsections. More technical details are provided in Appendix A.

3.1 Syntactic Graph based on Dependency Relation

Dependency parse tree can be regarded as a directed graph, where each edge represents the dependency relation between two nodes (words). It provides a hierarchical representation for plain text sentences and is considered to contain rich syntactic information. To make full use of the syntactic information contained in the dependency tree, we introduce a *syntactic graph builder* to convert the dependency tree (or say the raw dependency graph) into a *syntactic graph*, which is more compatible with graph neural networks and existing NAR-TTS structures.

The biggest challenge in extracting syntactic information with GNNs is the single-directed structure of the raw dependency graph, which denotes that the leaf node in the graph cannot obtain any information from other nodes during the graph aggregation. To handle this, inspired by previous works that exploit dependency relation in AR-TTS, we add a reverse edge for each directed edge in the dependency tree so that the information flow in the graph is bi-directional. Specifically, there could be *forward edges* from parent nodes to child nodes, which is consistent with the dependency tree, as well as *reversed edges* from child nodes to parent nodes. Then, we introduce our methods of constructing syntactic graphs with node embedding in specific languages.

Graph for English. To construct the syntactic graph for English text, we add *BOS* and *EOS* into the above-mentioned bi-directional graph and connect them with the first and last words of the input sentence, respectively. To be intuitive, we provide an example that transforms an English sentence into syntactic graph in Fig.3a, where *forward edges* are represented as solid black arrows and *reversed edges* are dashed black arrows. Then we consider the node representation in the constructed syntactic graph. Note that while TTS models typically use phoneme sequence as the input, the word is the fundamental unit in dependency parsing. To obtain the word-level node embedding, inspired by PortaSpeech, we adopt word-level average pooling to the phoneme encoding with word boundary information to generate the word encoding. As our node embedding is the latent encoding in the TTS model, it possesses valuable acoustic features for the TTS task and can be jointly optimized through backpropagation.

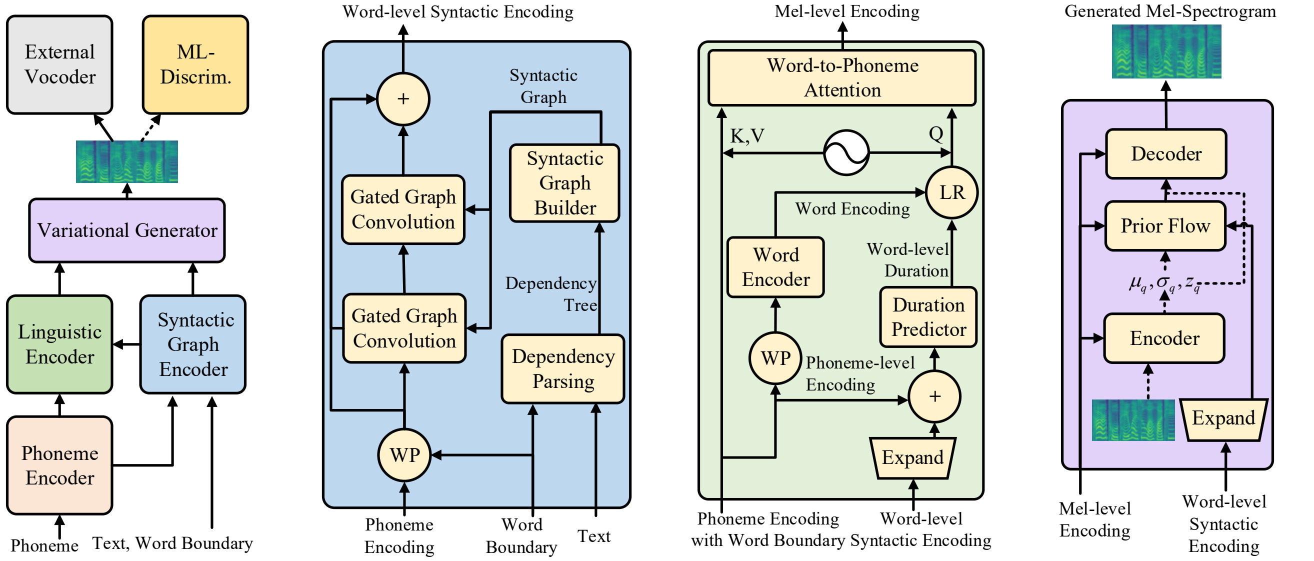
Graph for Chinese. As for the Chinese dataset, we make small adaptations. Different from English where the pronunciation of the word is directly decided by the phoneme, in Chinese the phoneme decides the pronunciation of the Chinese character, and the character decides the pronunciation of the word. To make the node representation more compatible with the Chinese pronunciation law, instead of extracting the word-level encoding as we design for English, we adopt character-level average pooling to generate the character encoding. To be coherent to the obtained character encoding, we extend the *syntactic graph* by expanding each word node into several Chinese character nodes, then use the first character node in each word to make the inter-word dependency connection, and other characters are sequentially connected according to the order in the word. Therefore, we additionally define two edges to represent the intra-word connection in *forward* and *reversed* directions, respectively. An intuitive example is shown in Fig.3b, where the green solid/dashed arrows denote the intra-word forward/reversed edges.

Graph for other languages. The syntactic graph for other languages can be constructed similarly. For instance, French and Spanish datasets can directly follow our approach for English, while Japanese datasets can use our graph construction method for Chinese.

3.2 Syntax-Aware Graph Encoder for Prosody Prediction

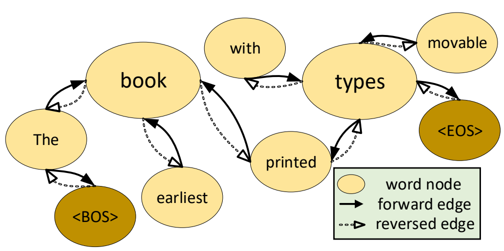
To learn the syntax-aware word representation from the input text, we design a *syntactic graph encoder* based on the *syntactic graph builder* and GNNs, which is shown in Fig. 2b. As illustrated in Sec.3.1, we process the input text with word boundary with the syntactic graph builder to generate a syntactic graph with heterogeneous edges (2 edges for English and 4 edges for Chinese), and in the meantime, the phoneme embedding is processed with word-level average pooling to formulate the node embedding in the syntactic graph. Now that the syntactic graph is equipped with learnable node embedding, the syntactic information is extracted through graph aggregation as follows: 1) we utilize two stacked Gated Graph Convolution layers with both 5 iterations to extract the long-term dependency in the graph; 2) the output of all preceding layers are summed up as the output syntactic word-level encoding, so as to assemble and reuse the word-level features from different receptive fields in the syntactic graph.

Then we consider embedding the extracted syntactic word

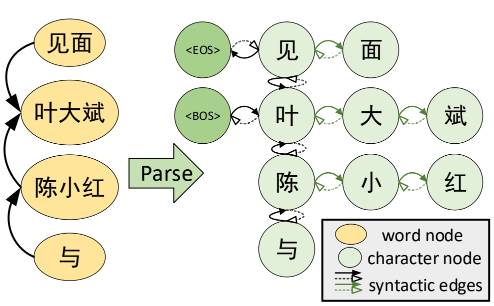


(a) SyntaSpeech (b) Syntactic Graph Encoder (c) Linguistic Encoder (d) Variational Generator

Figure 2: The overall structure for SyntaSpeech. In subfigure(a), ”ML-Discrim” denotes Multi-Length Discriminator in HiFiSinger. In subfigure (b), ”WP” denotes the word-level average pooling operation, and the ”Syntactic Graph Builder” is illustrated in Sec.3.1. In subfigure (c), ”LR” denotes the Length Regulator proposed in PortaSpeech. In subfigures (a) and (d), the dashed lines denote that the operations are only executed in the training phase.



(a) The graph built from the dependency tree in Fig.1.



(b) The graph built from a Chinese dependency tree.

Figure 3: Two examples of syntactic graph construction.

encoding into the TTS model. SyntaSpeech keeps main structures of PortaSpeech: a Transformer-based *linguistic encoder* to extract frame-level semantic representations with the help of a word-level duration predictor; a VAE-based *variational generator* with flow-based prior to synthesize the predicted mel-spectrogram. With these structures, PortaSpeech divides the prosody prediction (including duration, pitch, energy, etc.) into two sub-tasks: the duration predictor in linguistic encoder controls the timing in word-level; and in the variational generator, a flow-based enhanced prior distribution is introduced to predict the pitch, energy, and other prosody attributes. Based on the above insights, SyntaSpeech learns two individual syntactic graph encoders to extract syntactic features for duration prediction and other prosody attributes (e.g., energy and pitch) distribution modeling, respectively. To be specific, the extracted syntactic word encoding of the first graph encoder is expanded into phoneme-level and be fed into the duration predictor (as shown in Fig.2c), and the output of the second graph encoder is expanded into frame level as auxiliary features of the prior flow in variational generator (as shown in Fig. 2d).

3.3 Multi-Length Adversarial Training

The mel-spectrogram prediction of TTS models learned with mean square error (MSE) or mean absolute error (MAE) is generally challenged with blurry outputs. To handle this, PortaSpeech introduces a flow-based post-net to refine the predicted mel-spectrogram of the variational generator. Another common practice in handling the over-smoothing problem is to adopt the adversarial loss [Bińkowski *et al.*, 2020][Donahue *et al.*, 2021]. Following HiFiSinger [Chen *et al.*, 2020], we introduce a *multi-length discriminator* to distinguish between the output generated by the TTS model and the ground truth mel-spectrogram. Specifically, the variational generator is coupled with an ensemble of multiple CNN-based discriminators which evaluates the generated (true) spectrogram based on random windows of different lengths. Detailed structures can be found in Appendix A.2. Compared with using post-net in PortaSpeech, the benefits of multi-length adversarial training are twofold: 1) it can generate realistic spectrogram similar to post-net yet at a faster inference speed; 2) it can better capture unnatural slice in the generated sample and help improve the naturalness of word pronunciation.

4 Experiments

4.1 Experimental Setup

Datasets and Baselines. We evaluate SyntaSpeech on three datasets: 1) LJSpeech[1] [Ito and Johnson, 2017], a single-speaker database which contains 13,100 English audio clips with a total of nearly 24 hours speech; 2) Biaobei[2], a Chinese speech corpus consists of 10,000 sentences (about 12 hours) from a Chinese speaker; 3) LibriTTS[3] [Zen *et al.*, 2019], an English dataset with 149,736 audio clips (about 245 hours) from 1,151 speakers (We only use *train_clean360* and *train_clean100*). For computational efficiency, we first use the syntactic graph builder to process the raw text of the whole dataset to construct syntactic graphs and record them in the disk. We then load the mini-batch along with the pre-constructed syntactic graph during training and testing. The raw text is transformed into a phoneme sequence using an open-sourced grapheme-to-phoneme tool. The ground truth mel-spectrograms are generated from the raw waveform with the frame size 1024 and the hop size 256. We compare SyntaSpeech against two state-of-the-art NAR-TTS models: *PortaSpeech* and *FastSpeech 2*.

Model Configuration. SyntaSpeech consists of a phoneme encoder, a linguistic encoder, two syntactic graph encoders (with the same structures), a variational generator, and a multi-length discriminator. The phoneme encoder and linguistic encoder are based on multiple feed-forward Transformer blocks, and the variational generator uses the same structure in PortaSpeech. The multi-length discriminator is a lightweight CNN that consists of multiple stacked convolutional layers with batch normalization and treats the input spectrogram as images. We put more detailed model configurations in Appendix B.1.

Training and Evaluation. We train the SyntaSpeech on 1 Nvidia 2080Ti GPU with a batch size of 64 sentences. We use the Adam optimizer with $\beta_1 = 0.9$,$\beta_2 = 0.98$, $\epsilon = 10^{-9}$ and follow the same learning rate schedule in [Vaswani *et al.*, 2017]. It takes 320k steps for training until convergence. We use HiFi-GAN [Kong *et al.*, 2020] as the vocoder in LJSpeech and Biaobei, and use Parallel WaveGAN [Yamamoto *et al.*, 2020] as the vocoder in LibriTTS. We conduct MOS (mean opinion score) and CMOS (comparative mean opinion score) evaluations on the test set via Amazon Mechanical Turk. We analyze the MOS and CMOS in two aspects: prosody (naturalness of pitch, energy, and duration) and audio quality (clarity, high-frequency and original timbre reconstruction), and score MOS-P/CMOS-P and MOS-Q/CMOS-Q corresponding to the MOS/CMOS of prosody and audio quality. We put more details about the subjective evaluation in Appendix B.2.

[1]https://keithito.com/LJ-Speech-Dataset/

[2]https://www.data-baker.com/open_source.html

[3]http://www.openslr.org/60.

Method	LJSpeech	Biaobei	LibriTTS
GT	4.32 ± 0.09	4.43 ± 0.05	4.32 ± 0.07
GT (voc.)	4.26 ± 0.09	4.34 ± 0.05	4.29 ± 0.07
FastSpeech2	3.85 ± 0.12	3.75 ± 0.10	3.98 ± 0.08
PortaSpeech	4.01 ± 0.12	3.90 ± 0.10	4.06 ± 0.07
SyntaSpeech	$\mathbf{4.19 \pm 0.10}$	$\mathbf{4.12 \pm 0.07}$	$\mathbf{4.18 \pm 0.07}$

Table 1: MOS-P evaluation on three datasets.

Method	LJSpeech	Biaobei	LibriTTS
GT	4.26 ± 0.06	4.46 ± 0.05	4.25 ± 0.06
GT (voc.)	4.17 ± 0.08	4.33 ± 0.06	4.19 ± 0.08
FastSpeech2	3.94 ± 0.09	3.82 ± 0.09	3.95 ± 0.09
PortaSpeech	4.02 ± 0.08	4.05 ± 0.08	4.03 ± 0.10
SyntaSpeech	$\mathbf{4.13 \pm 0.08}$	$\mathbf{4.19 \pm 0.07}$	$\mathbf{4.10 \pm 0.08}$

Table 2: MOS-Q evaluation on three datasets.

4.2 Performance

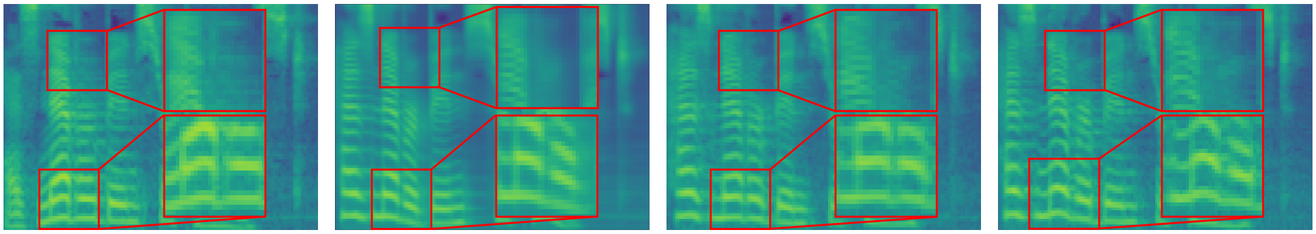
We compare the audio performance (MOS-P and MOS-Q) of our SyntaSpeech with other systems, including 1) *GT*, the ground truth audio; 2) *GT (voc.)*, where we first convert the ground truth audio into mel-spectrograms, and then convert the mel-spectrograms back to audio using external vocoders; 3) *FastSpeech2* [Ren *et al.*, 2021a]; 4) *PortaSpeech* [Ren *et al.*, 2021b]. We perform the experiments on three datasets as mentioned in Sec.4.1. The results are shown in Table 1 and 2. We observe that SyntaSpeech outperforms previous TTS models in both prosody (MOS-P) and audio quality (MOS-Q), which demonstrates its performance and robustness in multiple languages and multi-speaker TTS tasks. As our SyntaSpeech follows the variational generator in PortaSpeech, we perform a case study to demonstrate that SyntaSpeech could generate more natural audio than its baseline PortaSpeech, using a variety of latent variables of VAE. The result is put in Appendix C.1.

We then visualize the mel-spectrograms generated by the above systems in Fig.4. We can see that SyntaSpeech can generate mel-spectrograms with realistic pitch contours (which result in expressive prosody) and rich details in frequency bins (which result in natural sounds). In conclusion, our experiments demonstrate that SyntaSpeech could synthesize expressive and high-quality audio.

4.3 Ablation Studies

Syntactic Graph Encoder

We first analyze the effectiveness of the syntactic graph encoder to improve prosody from the perspective of training



(a) *GT* (b) *FastSpeech 2* (c) *PortaSpeech* (d) *SyntaSpeech*

Figure 4: Visualizations of the mel-spectrograms generated by different TTS systems. The corresponding text is *"has never been surpassed"*.

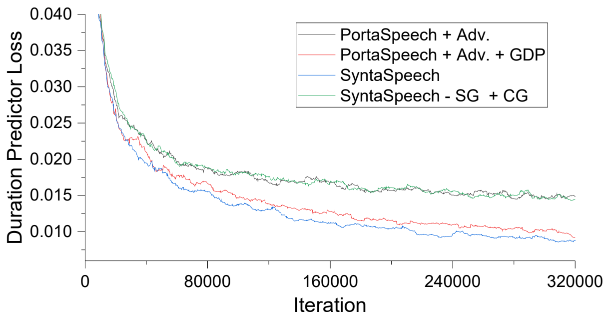


Figure 5: The duration predictor loss curves of several methods in LJSpeech. *Adv* denotes multi-length adversarial training, *GDP* denotes using graph encoder in duration predictor, *CG* denotes using complete graph instead of the syntactic graph (*SG*).

Settings	LJSpeech	Biaobei	LibriTTS
SyntaSpeech	0.000	0.000	0.000
- GDP	−0.131	−0.092	−0.119
- GPF	−0.069	−0.118	−0.059
- GDP - GPF	−0.152	−0.142	−0.168
- SG + CG	−0.160	−0.109	−0.188

Table 3: CMOS-P comparisons for ablation studies.

Settings	LJSpeech	Biaobei	LibriTTS
PortaSpeech	0.000	0.000	0.000
- PN + Adv.	0.071	0.088	0.050
SyntaSpeech	0.000	0.000	0.000
- Adv. + PN	−0.060	−0.166	−0.039

Table 4: CMOS-Q comparisons for ablation studies. *PN* denotes post-net in PortaSpeech, and *Adv* means our adversarial training.

objectives. The learning curves of duration predictor loss[4] in LJSpeech are shown in Fig.5. We observe that introducing a graph encoder in the duration predictor (*GDP*) could significantly improve the convergence. And SyntaSpeech, which is equivalent to *(PortaSpeech + Adv. + GDP + GPF)*, where *GPF* denotes using graph encoder in the prior flow, could further improve the performance. We also demonstrate that the improvement is brought by the syntactic information, as replacing the syntactic graph with the complete graph in SyntaSpeech leads to a similar curve to PortaSpeech. We put more objective evaluations in Appendix C.2.

We then perform CMOS evaluation to demonstrate the effectiveness of *syntactic graph encoder* in SyntaSpeech to improve prosody prediction. The results are shown in Table.3. We can see that CMOS-P drops when removing graph encoder in duration predictor (- *GDP*) or prior flow (- *GPF*), and replacing syntactic graph with complete graph (- *SG + CG*) leads to the largest CMOS-P degradation. A similar experiment that tests CMOS-Q can be found in Appendix C.3, in which we find that syntactic graph encoder has fewer impacts on the audio quality.

Adversarial Training

To demonstrate the effectiveness of adversarial training, we perform a CMOS test on PortaSpeech/SyntaSpeech with the multi-length adversarial training and the post-net. As can be seen in Table.4, both in PortaSpeech and our SyntaSpeech, multi-length adversarial training achieves better audio quality (CMOS-Q) than the flow-based post-net. We also compare the COMS-P, as can be found in Appendix C.4, in which we find that adversarial training also has slight improvements on the audio prosody.

[4]The duration predictor loss is the mean squared error between the logarithmic predicted word-level duration and the ground truth.

5 Conclusion

In this paper, we proposed SyntaSpeech, a syntax-aware and generative adversarial text-to-speech model. SyntaSpeech builds the syntactic graph from the dependency tree of the raw text, then extracts valuable syntactic information with graph convolution on the syntactic graph to improve the prosody prediction in the NAR-TTS model. We also introduced multi-length adversarial training to improve the audio quality and simplify the model architecture. We have demonstrated the performance and generalization ability of SyntaSpeech on three datasets (English, Chinese, and multi-speaker, respectively) and conducted comprehensive ablation studies to verify the effectiveness of each component in our model. For future work, we will explore the potential of syntax-aware models in other tasks, such as voice conversion and singing voice generation.

6 Acknowledgment

This work was supported in part by the National Key R&D Program of China under Grant No.2020YFC0832505, National Natural Science Foundation of China under Grant No.61836002, No.62072397, Zhejiang Natural Science Foundation under Grant LR19F020006.

References

[Bińkowski *et al.*, 2020] Mikołaj Bińkowski, Jeff Donahue, Sander Dieleman, Aidan Clark, Erich Elsen, Norman Casagrande, Luis C. Cobo, and Karen Simonyan. High fidelity speech synthesis with adversarial networks. In *ICLR*, 2020.

[Chen *et al.*, 2020] Jiawei Chen, Xu Tan, Jian Luan, Tao Qin, and Tie-Yan Liu. Hifisinger: Towards high-fidelity neural singing voice synthesis. *arXiv preprint arXiv:2009.01776*, 2020.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[Donahue *et al.*, 2021] Jeff Donahue, Sander Dieleman, Mikolaj Binkowski, Erich Elsen, and Karen Simonyan. End-to-end adversarial text-to-speech. In *ICLR*, 2021.

[Hirschberg and Rambow, 2001] Julia Hirschberg and Owen Rambow. Learning prosodic features using a tree representation. In *ECSCT*, 2001.

[Ito and Johnson, 2017] Keith Ito and Linda Johnson. The lj speech dataset. https://keithito.com/LJ-Speech-Dataset/, 2017. Accessed: 2022-06-07.

[Kim *et al.*, 2020] Jaehyeon Kim, Sungwon Kim, Jungil Kong, and Sungroh Yoon. Glow-tts: A generative flow for text-to-speech via monotonic alignment search. In *NIPS*, 2020.

[Kim *et al.*, 2021] Jaehyeon Kim, Jungil Kong, and Juhee Son. Conditional variational autoencoder with adversarial for end-to-end text-to-speech. In *ICML*, 2021.

[Kong *et al.*, 2020] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. In *NIPS*, 2020.

[Li *et al.*, 2016] Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. Gated graph sequence neural networks. In *ICLR*, 2016.

[Liu *et al.*, 2021] Rui Liu, Berrak Sisman, and Haizhou Li. Graphspeech: Syntax-aware graph attention network for neural speech synthesis. In *ICASSP*, 2021.

[Mishra *et al.*, 2015] Taniya Mishra, Yeon-jun Kim, and Srinivas Bangalore. Intonational phrase break prediction for text-to-speech synthesis using dependency relations. In *ICASSP*, 2015.

[Peng *et al.*, 2020] Kainan Peng, Wei Ping, Zhao Song, and Kexin Zhao. Non-autoregressive neural text-to-speech. In *ICML*, 2020.

[Ping *et al.*, 2018] Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan O Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, and John Miller. Deep voice 3: Scaling text-to-speech with convolutional sequence learning. In *ICLR*, 2018.

[Ren *et al.*, 2019] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech: fast, robust and controllable text to speech. In *NIPS*, pages 3171–3180, 2019.

[Ren *et al.*, 2021a] Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech 2: Fast and high-quality end-to-end text to speech. In *ICLR*, 2021.

[Ren *et al.*, 2021b] Yi Ren, Jinglin Liu, and Zhou Zhao. Portaspeech: Portable and high-quality generative text-to-speech. In *NIPS*, 2021.

[Sun *et al.*, 2020] Aolan Sun, Jianzong Wang, Ning Cheng, Huayi Peng, Zhen Zeng, and Jing Xiao. Graphtts: graph-to-sequence modelling in neural text-to-speech. In *ICASSP*, 2020.

[Sun *et al.*, 2021] Aolan Sun, Jianzong Wang, Ning Cheng, Huayi Peng, Zhen Zeng, Lingwei Kong, and Jing Xiao. Graphpb: Graphical representations of prosody boundary in speech synthesis. In *SLT*, 2021.

[van den Oord *et al.*, 2016] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. In *SSW*, 2016.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

[Wang *et al.*, 2017] Yuxuan Wang, R. J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Z. Chen, Samy Bengio, Quoc V. Le, Yannis Agiomyrgiannakis, Robert A. J. Clark, and Rif A. Saurous. Tacotron: Towards end-to-end speech synthesis. In *INTERSPEECH*, 2017.

[Yamamoto *et al.*, 2020] Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP*, 2020.

[Zen *et al.*, 2019] Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. Libritts: A corpus derived from librispeech for text-to-speech. *arXiv preprint arXiv:1904.02882*, 2019.

[Zhou *et al.*, 2021] Yixuan Zhou, Changhe Song, Jingbei Li, Zhiyong Wu, and Helen Meng. Dependency parsing based semantic representation learning with graph neural network for enhancing expressiveness of text-to-speech. *arXiv preprint arXiv:2104.06835*, 2021.